НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено
В. о. завідувача кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 "Системний аналіз"
на тему: «Алгоритм видобутку інформації з неструктурованих текстових джерел»

Виконав (-ла):
Студент (-ка) IV курсу, групи КА-61
Баздирев Антон Андрійович _____

Керівник:
доцент, к.ф.-м.н. Каніовська Ірина Юріївна _____

Консультант з економічного розділу:
доцент, к.е.н., Шевчук Олена Анатоліївна _____

Консультант з нормоконтролю:
доцент, к.т.н., Коваленко Анатолій Єпіфанович _____

Рецензент:
доцент, к.ф.-м.н. Буценко Юрій Павлович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 "Системний аналіз"
Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри
_____ О.Л. Тимощук
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Баздиреву Антону Андрійовичу

1. Тема роботи «Алгоритм видобутку інформації з неструктурованих текстових джерел», керівник роботи Каніовська І. Ю. доцент, к.ф.-м.н, затверджені наказом по університету від «_25_»_травня__ 2020 р. №1143-с

2. Термін подання студентом роботи_____

3. Вихідні дані до роботи _____

_____

4. Зміст роботи _____

_____

_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А., доцент	20.04.20	04.05.20

7. Дата видачі завдання _____

Календарний план

№ з	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка

Студент _____ Баздирев А. А.
(підпис) (ініціали, прізвище)

Керівник роботи _____ Каніовська І. Ю.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Дипломна робота містить: 90 с., 43 рис., 13 табл., 2 дод., 13 джерел.

ОБРОБКА ПРИРОДНОЇ МОВИ, ГЛИБОКЕ НАВЧАННЯ, ТРАНСФЕРНЕ НАВЧАННЯ, ВИДОБУТОК ІНФОРМАЦІЇ, ТРАНСФОРМЕРИ

Об’єкт дослідження – неструктуровані текстові дані, зокрема тексти з веб-сайтів та офіційної документації.

Предмет дослідження - рекурентні нейронні мережі, нейронні мережі трансформери.

Мета роботи – описати та заформалізувати загальний підхід до видобутку довільної інформації з неструктурованих текстових джерел за допомогою методів глибокого навчання та створити систему розпізнавання та видобутку з веб-сайтів компаній офіційних імен та адрес.

Результатом роботи є створена та розгорнена в кластері в хмарі AWS система, що розпізнає офіційні імена компанії та адреси в довільних текстових документах та веб-сайтах. У роботі застосовано методи глибокого та трансферного начання. В якості нейронних мереж кодувальників використано BERT та LSTM.

Актуальність дослідження полягає в тому, що алгоритми роботи з довільними текстовими даними наразі знаходять широке застосування в компаніях, що займаються стратегічним консультуванням, аналізом ризиків та створенням загальних всеохоплюючих комерційних баз знань та експертних систем.

ABSTRACT

Thesis: 90 pp., 43 fig., 13 tables, 2 appendix, 13 sources.

NATURAL LANGUAGE PROCESSING, DEEP LEARNING, TRANSFER LEARNING, DATA EXTRACTION, TRANSFORMER NETWORKS

The object of research - unstructured textual data, including texts from websites and official documentation.

The subject of research - recurrent neural networks, neural networks transformers.

The purpose of the work is to describe and formalize the general approach to the information extraction from unstructured text sources using deep learning models and to create a system for recognition and extraction of official names and addresses from the websites of companies.

The result is a deployed system, which extracts the official company names and addresses in arbitrary text documents and websites. The methods of deep and transfer learning are used in the work. BERT and LSTM were used as encoders.

The relevance of the study is that algorithms for natural language procesing are widely used in companies engaged in strategic consulting, risk analysis and the creation of common comprehensive commercial knowledge bases and expert systems.

ЗМІСТ

ПОСТАНОВКА ЗАДАЧІ

1. Побудувати загальний підхід до видобутку інформації з неструктурованих текстових джерел.
2. Застосувати підхід до реальної задачі про видобуток офіційних імен та адрес з розбиттям на окремі компоненти з сайтів компаній.
3. Дослідити вплив застосування різноманітних технік трансферного навчання для реальної задачі.
4. Створити та розгорнути систему на основі мікросервісної архітектури для вирішення вищеописаної задачі.

ВСТУП

На сьогоднішні день, в еру діджиталізації та стрімкого розвитку мережевих технологій дуже важливо своєчасно отримувати максимально повну інформацію в структурованому вигляді. Створення алгоритму автоматичного видобутку інформації в цьому ключі може бути дуже корисним. Поточний розвиток технологій глибокого та трансферного навчання дозволяє створити такий алгоритм.

Актуальність роботи полягає у наступному: такий алгоритм та методологія його побудови може знайти широке застосування для компаній, що займаються стратегічним консультуванням, аналізом ризиків та створенням загальних всеохоплюючих комерційних баз знань та експертних систем.

Мета дослідження: Аналіз існуючих алгоритмів та методів обробки природної мови, глибоких нейронних мереж та стратегій і методів трансферного навчання та ідея побудова загальної моделі для видобутку довільної заданої інформації з неструктурованих текстових джерел. Експериментальні дослідження для конкретної задачі різних методів обробки природної мови.

Методи та об’єкти дослідження: метод максимальної правдоподібності, рекурентні нейронні мережі, нейронні мережі трансформери.

РОЗДІЛ 1 МОДЕЛІ ГЛИБОКОГО НАВЧАННЯ ТА ОГЛЯД ЇХ ЗАСТОСУВАННЯ ДЛЯ РОБОТИ З ПРИРОДНОЮ МОВОЮ

1.1 Вступ

В цьому розділі розглядаються основні алгоритми, поняття та методи, які пов’язані з сучасними моделями глибокого навчання для обробки природної мови. Описуються загальні ідеї різних архітектур глибоких нейронних мереж, а також методів та стратегій трансферного навчання. Проводиться опис основних операцій в рекурентних нейронних мережах та трансформерах.

1.2 Токенізація

Токенізація - це ключова базова операція в обробці природної мови, яка полягає в розділенні суцільного тексту на базові структурні одиниці - токени. Як правило, виділяють токенізацію на рівні речень - розділення великого тексту на абзаци або речення зазвичай за допомогою регулярних виразів, що розділяють текст через пунктуацію, пробіли чи спеціальні символи; та токенізацію на рівні слів - розділення речення на словосполучення, окремі слова чи навіть частинки від слів. Зараз дуже поширеною є токенізація, що допускає розбиття слів на їх частини й залежить від словника, наприклад токенізація для моделі BERT (рисунок 1.1). Оскільки моделі будуються над скінченними словниками, то є сенс токенізувати текст так, щоб токени в результаті були наявні в словнику.

```
tokenizer.tokenize("don't be so judgmental")
```

```
['don', "'", 't', 'be', 'so', 'judgment', '##al']
```

Рисунок 1.1

1.3 Ембеддінги

Моделі машинного та глибокого навчання працюють з векторами, тому для їх застосування необхідно певним чином перетворити токени (слова) на вектори. Нехай є деякий скінченний словник, що містить n токенів. Кожному токену можна співставити одиничний вектор, в якому 1 стоїть в координаті, що відповідає індексу токена в словнику. Використання такого роду векторів є неефективним, оскільки розмірність векторів є дуже великою й всі токени розташовані на однаковій відстані один від одного. Тому вводиться поняття ембеддінгу - відображення токенів в векторний простір відносно малої розмірності (зазвичай до 300), при чому таке відображення будується так, щоб схожі слова мали вектори, косинус кута між якими близький до 1.

Формально кажучи : Нехай S − словник розмірності n; $S = \{e_i \mid i \text{ від } 1 \text{ до } n\}$;

E − ембеддінг розмірності m над S; $E : \mathbb{R}^n \longrightarrow \mathbb{R}^m$

v_i - вектор, що відповідає токену з індексом і зі словника; $v_i = Ee_i$

Приклад: знаходження векторів для слів [‘dog’, ‘spoon’, ‘scissors’, ‘guitar’] ембеддінгу розмірності 3 над словником розміності 81 - рисунок 1.2

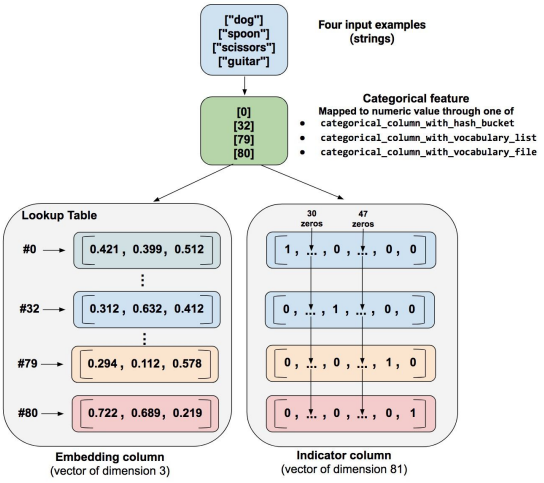


Рисунок 1.2

1.4 LSTM мережа

Оскільки слова в природній мові йдуть послідовно одне за одним, то доцільно розглядати текст як послідовність і використати модель рекурентної архітектури. Тут будемо розглядати одну з найефективніших рекурентних моделей - LSTM (long short-term memory, дослівно - довга короткострокова пам’ять) [1]. LSTM мережа була спеціально розроблена для усунення проблеми забування довгострокових залежностей. Наведемо загальну схему цієї рекурентної мережі на рисунку 1.3. Як xt позначаємо вхідний вектор на позиції t, як ht позначаємо вихідний вектор мережі на позиції t. В задачах обробки природної мови xt - ембеддінг вектор t-го токену вхідного тексту. При чому тренування ембеддінгу може відбуватися при тренуванні всієї мережі або можна використовувати вже попередньо натреновані ембеддінг моделі, зокрема glove або fasttext.

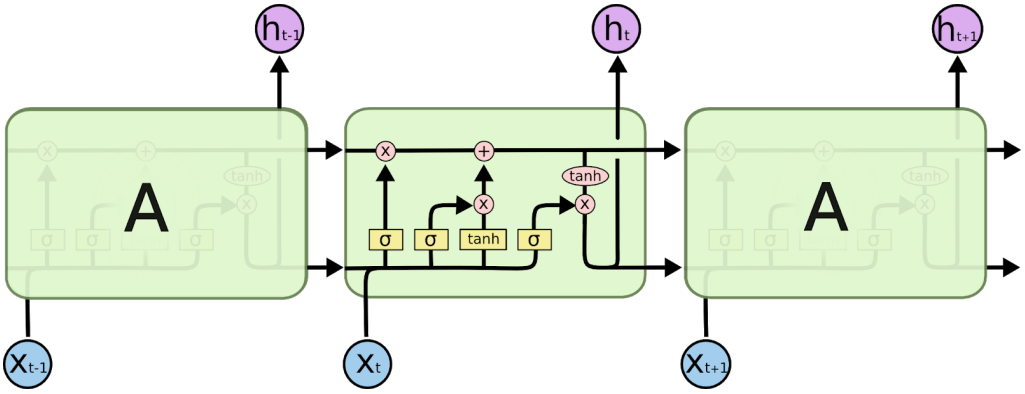
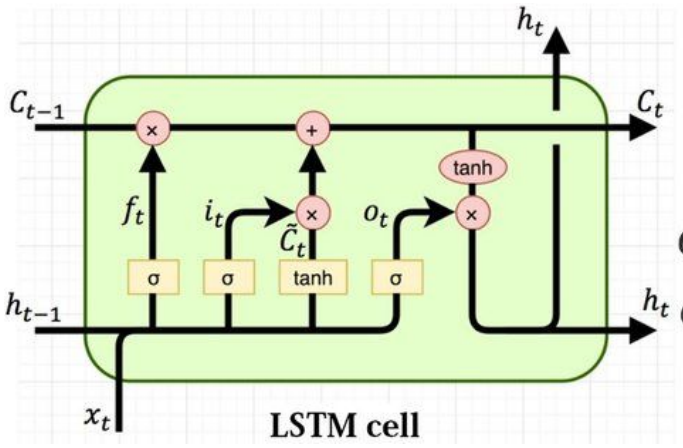


Рисунок 1.3

Також наведемо більш детальну схему з формулами власне LSTM-клітини на рисунку 1.4



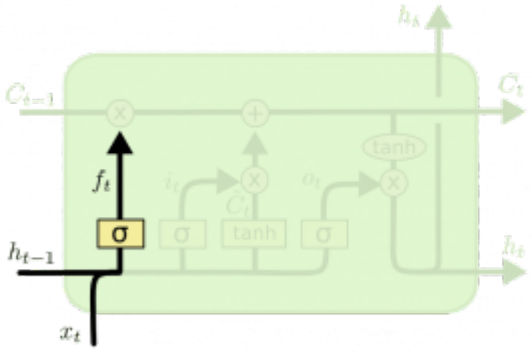
$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Рисунок 1.4

Одним з основних понятть в LSTM є стан клітини - c_t. Вектор c_t по суті є агрегованим в одному векторі представленням всієї вхідної послідовності векторів $x_1 ... x_t$. Для контролю c_t LSTM має три ключових механізми: забування, збереження та оновлення стану.

На етапі забування вирішується яку інформацію (значення координати вектора c_t) потрібно “забути” і наскільки (Рисунок 1.5). f_t - вихід шару забування, W_f (матриця) та b_f (вектор) - параметри шару забування, які оцінюються під час тренування моделі, x_t - вхідний вектор на поточному кроці, h_{t-1} - вихідний вектор з попереднього кроку, σ - сигмоїд. $\sigma(x) = \frac{1}{1+e^{-x}}$

f_t покоординатно перемножується на c_{t-1} і оскільки всі координати c_{t-1} є (0, 1), то таке множення реалізує механізм “забування”.

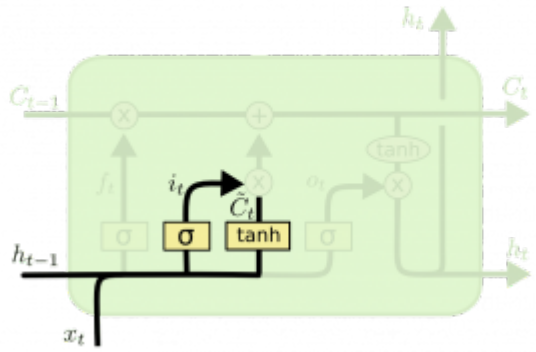


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

Рисунок 1.5

Наступною виконується операція збереження стану (Рисунок 1.6). Сигмоїдальний шар i_t відповідає за те, які координати стану клітини потрібно

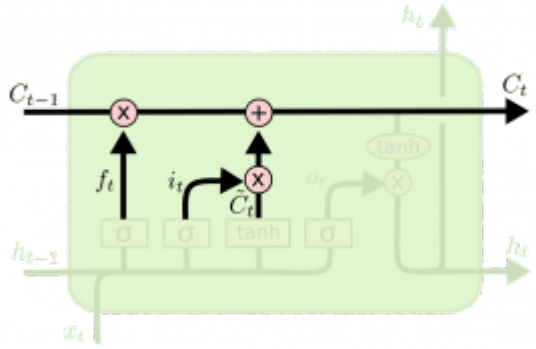
буде оновити, а шар C_t обчислює нові значення, які йдуть в наступний шар оновлення. W_i, b_i , W_C, b_C- параметри шару, які оцінюються під час тренування моделі, x_t - вхідний вектор на поточному кроці, h_{t-1} - вихідний вектор з попереднього кроку, σ - сигмоїд.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Рисунок 1.6

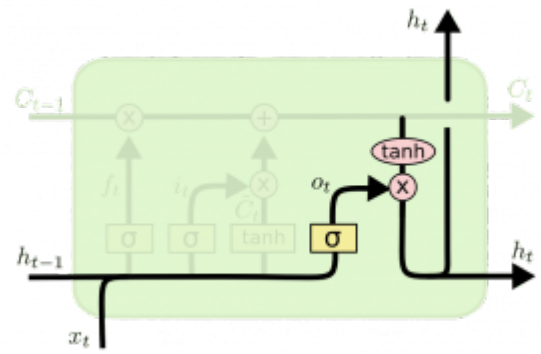
Потім виконується операція оновлення стану (Рисунок 1.7). Виконується покоординатне перемноження результуючих векторів шару збереження, та покоординатне додавання результату до виходу шару забування.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Рисунок 1.7

Зрештою, можемо обчислити значення вихідного вектору h_t (Рисунок 1.8). W_o та b_o - параметри вихідного шару, які оцінюються під час тренування моделі, x_t - вхідний вектор на поточному кроці, h_{t-1} - вихідний вектор з попереднього кроку, σ - сигмоїд, tanh - гіперболічний тангенс.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Рисунок 1.8

Як і в багатьох інших випадках, для навчання рекурентних мереж використовуються градієнтні методи. В простих рекурентних нейронних мережах існує проблема “затухання” градієнту [2]. Дослідимо, що відбувається з градієнтами параметрів LSTM. Нехай при поточній довжині послідовності k є довільний критерій E_k, що оптимізується. Розглянемо градієнти для W_f, W_i , W_C та W_o.

$$\frac{\partial Ek}{\partial Wf} = \frac{\partial Ek}{\partial h(t)}\frac{\partial c(t)}{\partial c(t-1)} \cdots \frac{\partial c(2)}{\partial c(1)}\frac{\partial c(1)}{\partial Wf} \qquad \frac{\partial Ek}{\partial Wi} = \frac{\partial Ek}{\partial h(t)}\frac{\partial c(t)}{\partial c(t-1)} \cdots \frac{\partial c(2)}{\partial c(1)}\frac{\partial c(1)}{\partial Wi}$$

$$\frac{\partial Ek}{\partial WC} = \frac{\partial Ek}{\partial h(t)}\frac{\partial c(t)}{\partial c(t-1)} \cdots \frac{\partial c(2)}{\partial c(1)}\frac{\partial c(1)}{\partial WC} \qquad \frac{\partial Ek}{\partial Wo} = \frac{\partial Ek}{\partial h(t)}\frac{\partial c(t)}{\partial c(t-1)} \cdots \frac{\partial c(2)}{\partial c(1)}\frac{\partial c(1)}{\partial Wo}$$

Як бачимо, загальний вигляд цих градієнтів є однаковим, тому надалі будемо використовувати запис $\frac{\partial Ek}{\partial W}$, $\otimes-$ покоорд. множення, $\oplus-$ покоорд. *с*ума

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k} \cdots \frac{\partial c_2}{\partial c_1}\frac{\partial c_1}{\partial W}$$

$$= \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}\frac{\partial c_t}{\partial c_{t-1}}\right)\frac{\partial c_1}{\partial W}$$

$$c_t = c_{t-1} \otimes \sigma\left(W_f \cdot [h_{t-1}, x_t]\right) \oplus$$
$$tanh\left(W_c \cdot [h_{t-1}, x_t]\right) \otimes \sigma(W_i \cdot [h_{t-1}, x_t])$$

$$\frac{\partial c_t}{\partial c_{t-1}} = \frac{\partial}{\partial c_{t-1}}[c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t]$$

$$= \frac{\partial}{\partial c_{t-1}}[c_{t-1} \otimes f_t] + \frac{\partial}{\partial c_{t-1}}[\tilde{c}_t \otimes i_t]$$

$$= \frac{\partial f_t}{\partial c_{t-1}} \cdot c_{t-1} + \frac{\partial c_{t-1}}{\partial c_{t-1}} \cdot f_t + \frac{\partial i_t}{\partial c_{t-1}} \cdot \tilde{c}_t + \frac{\partial \tilde{c}_t}{\partial c_{t-1}} \cdot i_t$$

$$\frac{\partial c_t}{\partial c_{t-1}} = \sigma'(W_f \cdot [h_{t-1}, x_t]) \cdot W_f \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot c_{t-1}$$

$$+ f_t$$

$$+ \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot \tilde{c}_t$$

$$+ \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot i_t$$

Позначимо:

$$A_t = \sigma'(W_f \cdot [h_{t-1}, x_t]) \cdot W_f \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot c_{t-1}$$

$$B_t = f_t$$

$$C_t = \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot \tilde{c}_t$$

$$D_t = \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot i_t$$

Таким чином, маємо:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{t=2}^{k} [A_t + B_t + C_t + D_t]\right) \frac{\partial c_1}{\partial W}$$

Зауважимо, що $B_t = f_t > 0$

Нехай, $\sum_{t=1}^{k} \frac{\partial E_t}{\partial W} \to 0$ Але $\exists\, k+1 \mathrm{i}\, f(k+1)$ такі, $\frac{\partial E_{k+1}}{\partial W} \not\to 0$ що

Тоді, $\sum_{t=1}^{k+1} \frac{\partial E_t}{\partial W} \not\to 0$

Отже, “затухання” градієнта не відбувається.

1.5 Трансферне навчання

На сьогодні трансферне навчання є одним із найбільш популярних напрямів у сфері штучного інтелекту. В найбільш поширеному сенсі його суть полягає в тому, що модель спочатку навчають на деякому дуже великому обсягу даних (pretraining), можливо, з іншої предметної області, можливо, для однієї або декількох інших задач. А потім дотреновують верхні шари для конкретної задачі (finetuning). При чому для цього зазвичай вже немає необхідності у великій тренувальній вибірці [3].

Формально задача трансферного навчання виглядає так [4]:

Нехай $\exists\ D_1 -$ деяка предметна область. $D_1 = \{X,\ P(x)\},$ де $X -$ простір факторів, $P(x)\ -$ задає розподіл ймовірностей; $x = \{x_1, x_2 \ldots x_n\} \in X$

$\exists\ T_1 -$ деяка задача в $D_1,$ яка складається з $Y -$ простору цільових змінних, та f_1 предиктивної функції, яка отримана шляхом оптимізації деякого критерію оптимальності $L_1 : Y \times Y \rightarrow R$ на вибірці $\{x_i, y_i\}$, де $x_i \in X, y_i \in Y$

$\exists\ D_2 -$ інша предметна область та T_2 задача в D_2 і критерій оптимальності L_2.

Задача трансферного навчання полягає в тому, щоб покращити значення критерію L_2 для задачі T_2 за допомогою використання D_1 та T_1. В загальному випадку $D_1 \neq D_2$, $T_1 \neq T_2$, $L_1 \neq L_2$.

Історично успіхи в трансферному навчанні розпочалися з напряму комп'ютерного зору. Всі сучасні передові архітектури мають реалізацію моделі, натренованої на задачу класифікації на ImageNet - найвідомішому і одному з найбільших наборів різнопланових картинок у відкритому доступі. Потім при застосуванні таких моделей для власних потреб зазвичай замінюють верхній шар глибокої нейронної мережі й донавчають для конкретної задачі, при чому не обов'язково класифікації.

В предметній області обробки природної мови виділяють різні класи задач трансферного навчання [5] (Рисунок 1.9).

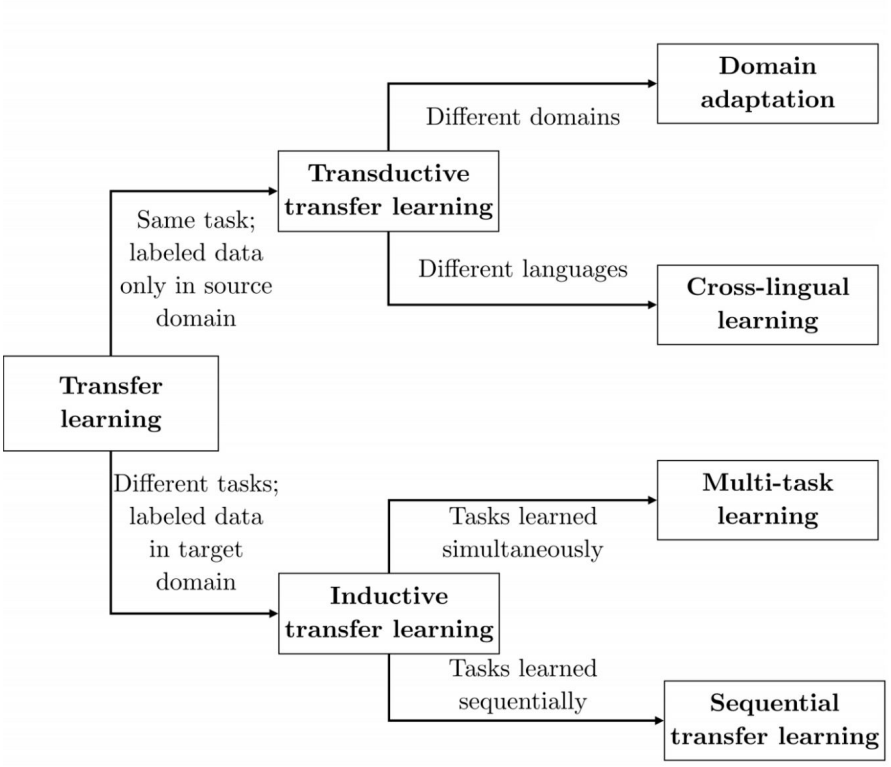


Рисунок 1.9

- Domain adaptation - випадок однакової задачі, але різних предметних областей. Може не завжди добре працювати, оскільки предметні області можуть бути зовсім різними й тому, через лінгвістичні особливості. Перенесення моделі на іншу предметну область може не бути успішною.
- Cross-lingual learnign - випадок перенесення моделі для схожої предметної області і тієї ж задачі, але з вхідними даними іншою мовою. Добре працює, якщо правила побудови речень і висловлень в мовах схожі.
- Multi-task learning - навчання однієї моделі для вирішення декількох задач одночасно зазвичай в одній предметній області.
- Sequential transfer learning - послідовне навчання моделі для вирішення різних задач в одній предметній області.

Найбільш поширеною задачею для попереднього навчання моделей є так звана задача маскованого мовного моделювання (masked language modeling). Для цієї задачі підійде будь-яка текстова вибірка й суть її полягає в тому, що в тексті випадковим чином деякі токени замінюють на “маски” і модель навчають так, щоб вона правильно відновлювала (класифікувала) пропущені токени.

Для навчання моделі в якості цільового критерію беруть максимізацію логарифму правдоподібності: $L = \sum_{i=1} log(P(v_i | v_{i-m} \ldots v_{i-1}; \Theta))$;

де v_i - масковане слово, m - розмір ковзного вікна; Θ - параметри моделі, що оцінюються. Власне оптимізація відбувається зазвичай за допомогою модифікації методу градієнтного спуску.

1.6 Архітектура типу трансформер

Трансформер - це нейронна мережа глибокого навчання, представлена в 2017 році [6] перш за все для задач обробки природної мови. Моделі, що базуються на цій архітектурі є передовими і демонструють найкращі результати за метриками в більшості відповідних задач, зокрема класифікації, машинного перекладу, сумаризації та інших. Хоча трансформери й створені для обробки даних у вигляді послідовностей, на відміну від звичайних рекурентних нейронних мереж, трансформер має набагато більше можливостей для обробки вхідних даних паралельно, а не послідовно, що може пришвидшувати тренування моделі. В загальному випадку, трансформер складається з двох частин: кодувальника, який робить представлення послідовності слів у вигляді векторної послідовності, та декодувальника - відображає векторну послідовності в послідовність токенів (слів). В основі обох цих частин лежить механізм self-attention. Також більшість трансформероподібних моделей мають попередньо натреновані багатомовноі реалізації та реалізації для різних мов

окремо, які можна знайти в tensorflow-hub (tensorflow реалізації) та pytorch-transformers (відповідно pytorch). Окремим напрямом в розробці, зокрема трансформер моделей, є дистиляція - створення моделей з меншою кількістю параметрів на основі великих моделей без втрати якості.

1.6.1. Self-Attention механізм

Self-Attention механізм [6] є ключовим в будь-якій трансформероподібній архітектурі. Ще до винаходу self-attention в глибоких нейонних мережах для обробки послідовностей використовувався attention механізм, який реалізовував можливість подачі на кожному кроці на вхід мережі лінійну комбінацію лінійних перетворювань вхідних векторів динамічно, замість просто звичайного виходу з попереньої рекурентної клітини, що зазвичай покращувало результат. На рисунку 1.10 зображено візуалізацію роботи self-attention механізму.

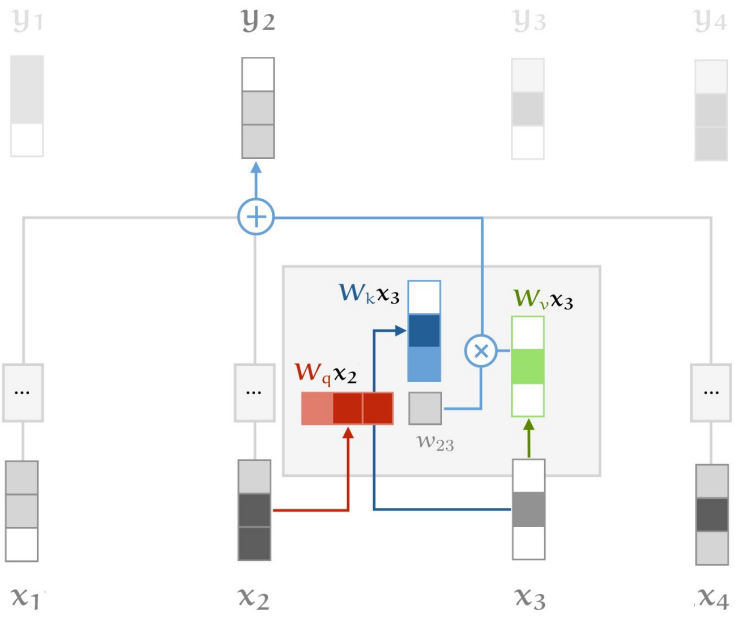


Рисунок 1.10

$$\mathbf{q}_i = \boldsymbol{W}_q \mathbf{x}_i \qquad \mathbf{k}_i = \boldsymbol{W}_k \mathbf{x}_i \qquad \boldsymbol{v}_i = \boldsymbol{W}_v \mathbf{x}_i$$

$$w'_{ij} = \mathbf{q}_i{}^{\mathsf{T}} \mathbf{k}_j$$

$$w_{ij} = \mathrm{softmax}(w'_{ij})$$

$$\mathbf{y}_i = \sum_j w_{ij} \boldsymbol{v}_j \,.$$

$\{x_1, x_2 ... x_n\}$ - послідовність вхідних векторів

$\{y_1, y_2 ... y_n\}$ - послідовність вихідних векторів - результат роботи механізму

W_k, W_q, W_v - так звані key, querry та value матриці, всі розмірності nxn, де n розмірність x_n, вони є параметрами моделі, які оцінюються в процесі тренування нейронної мережі.

softmax - узагальнення логістичної функції $softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$, де z - вектор розмірності K. Зауважимо, що $\sum_{j=1}^{K} softmax(z)_j = 1$

Для того, щоб ваги не залежали від розмірності векторів виконують нормування:

$$w'_{ij} = \frac{\mathbf{q}_i{}^{\mathsf{T}} \mathbf{k}_j}{\sqrt{k}}$$

де k - розмірність x_n

Таким чином, операція self-attention в матричному вигляді:

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{n}})\mathbf{V}$$

1.6.2. Multi-head self-attention та базовий блок трансформера

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, ..., \mathrm{head}_h) W^O$$
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Схема Multi-head self-attention модулю зображена на рисунку 1.11

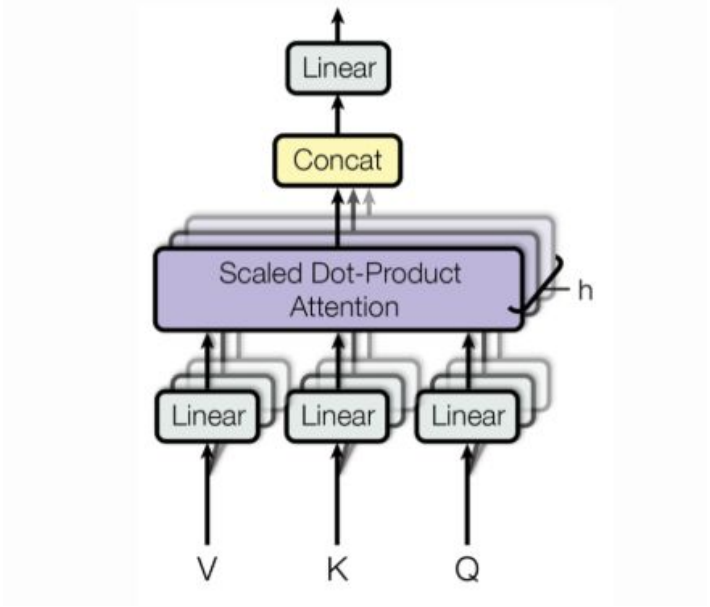


Рисунок 1.11

По суті в цьому модулі є n - self-attention голів, результуючі вектори яких конкатенуються і проектуються в простір початкової розмірності вхідних векторів за допомогою оператора W^O: $\mathbb{R}^{nk} \longrightarrow \mathbb{R}^k$, де k - розмірність вхідних векторів [6].

Зобразимо базову схему блоку моделі трансформер на рисунку 1.12

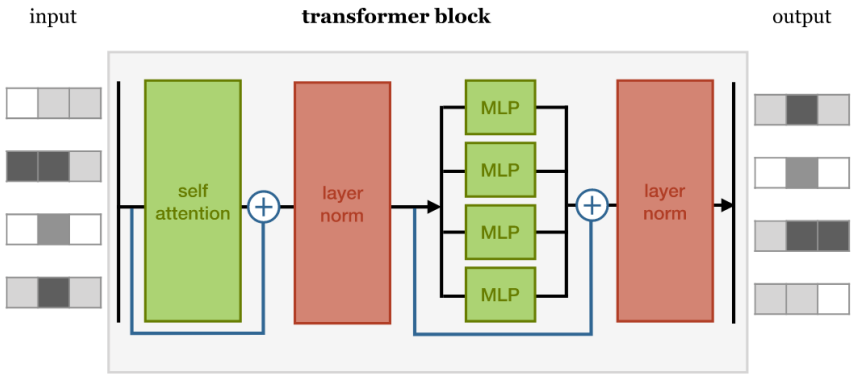


Рисунок 1.12

На рисунку 1.12 self-attention позначає multi-head self attention head в загальному випадку з довільною кількістю голів. Сині лінії з операцією $\oplus$ позначають residual оператор, який працює таким чином: нехай є деяка векторна функція f: $\mathbb{R}^n \longrightarrow \mathbb{R}^n$, тоді residual(f)(x) = f(x) $\oplus$ x, де $\oplus$ - покоординатна сума. Блоки MLP позначають звичайну багатошарову feed-forward нейронну мережу.

Детальніше будемо розглядати схему Layer Normalization шару:

Нехай x - деякий батч при навчанні моделі стохастичним градієнтним спуском чи деякою його модифікацією. x - матриця nxm, де n - кількість елементів в батчі, а m - розмірність кожного елемента.

Тоді: $\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij}$; $\sigma_i^{\,2} = \frac{1}{m}\sum_{j=1}^{m}(x_{ij}-\mu_i)^2$

$\widehat{x_{ij}} = \frac{x_{ij}-\mu_i}{\sqrt{\sigma_i^2+\varepsilon}}$; ε – деяка величина порядку 10^{-5}, щоб уникнути ділення на 0

Маємо $LayerNormalization(x)_i = \gamma\widehat{x_i} + \beta$, де γ, β - параметри цього шару, які оптимізуються при навчанні моделі.

1.6.3. BERT

BERT (Bidirectional Encoder Representations from Transformers) [7][8] - це кодувальник, який складається з послідовних 2 - 24 блоків-трансформерів (2 - BERT Tiny, 24 - BERT Large), архітектура яких була описана вище. Ця модель і її похідні (ALBERT, RoBERTa, і т.д.) демонструють провідні результати на всіх поточних бенчмарках в напрямі обробки природної мови в різних задачах. Офіційно в публічному доступі є багатомовні моделі, та окремі моделі для різних мов, які були попередньо натреновані для задачі маскованого мовного моделювання на великих текстових даних, таких як Wikipedia.

1.7 Висновки до розділу 1

В цьому розділі розглянуто основні типи глибоких нейронних мереж, які наразі активно застосовують до задач обробки природної мови та їх основні складові та механізми. Одним із ключових базових понятть є ембеддінг - відображення токена (слова) у вектор відносно невеликої розмірності. Важливими елементами в клітинах LSTM нейронної мережі є такі шари (гейти): забування - вирішує яку інформацію (значення координати вектора ct) потрібно “забути” і наскільки, збереження - відповідає за те, які координати стану клітини потрібно буде зберігти, а які оновити та оновлення - власне оновлює стан клітини LSTM. Також проаналізовано архітектуру LSTM на предмет проблеми “затухання” градієнта при навчанні й отримано те, що ця проблема даній архітектурі радше не притаманна, на відміну від простої рекурентної нейронної мережі. Також розглянуто різні задачі та методи трансферного навчання в цілому та конкретні види підходів для напрямку обробки природної мови. Для розуміння ідей більшості сучасних провідних моделей цього напрямку важливим є розуміння механізму self-attention, оскільки здебільшого на ньому будується архітектура всіх моделей типу трансформер, а також таких речей як Layer Normalization та residual зв’язків, які теж були розглянуті та описані в цьому розділі.

РОЗДІЛ 2 ПОБУДОВА МОДЕЛІ ВИДОБУТКУ ІНФОРМАЦІЇ З ТЕКСТОВИХ ДЖЕРЕЛ ТА ЕКСПЕРИМЕНТИ З НАЯВНИМ НАБОРОМ ДАНИХ

2.1 Вступ

В цьому розділі буде запропонована формалізація задачі видобутку інформації з довільних неструктурованих текстових джерел а також побудова загального підходу до її вирішення за допомогою моделей глибокого та трансферного навчання. Буде детально описано наявний набір даних, на якому проведені експерименти з конкретними реалізаціями запропонованого загального підходу, зокрема буде порівняно використання таких моделей, як LSTM та BERT в якості однієї зі складових системи, а також підходи з адаптації предметної області (domain adaptation) та послідовного трансферного навчання (sequential transfer learning).

2.2 Формалізація задачі та побудова загального підходу

2.2.1 Формалізація постановки задачі

Нехай є деякий скінченний словник токенів $V = \{w_0, w_1,..w_n\}$. Нехай, маємо деякі типи текстової інформації $\{c_1,..c_k\}$, відповідну інформацію релевантну ним ми хочемо отримати в структурованому вигляді з довільного джерела T. Під отриманням інформації в структурованому вигляді маємо на увазі створення таких множин C_j (можливо пустих):

$C_j = \{w \mid w \in T \wedge r(w, c_j) \wedge w \in V\}$, при чому

$\forall\, w \in T,\ \forall\, j = 1, k\ \neg(w \in C_j) \rightarrow (r(w, c_j) = 0)$, де

$r(w, c) = 1$ при "w релевантно"; та $r(w, c) = 0$, при "w не релевантно c"

Розглянемо приклад: c_1 = ”країна”, c_2 = ”ім’я”, c_3 = “професійна діяльність”;

T = “Петро та Леся повернулися зі своєї відпустки в Іспанії, яка пройшла чудово!”

В результаті матимемо: C_1 = {‘Іспанії’}; C_2 = {‘Петро’, ‘Леся’}; C_3 = {}

2.2.2 Задача розмітки послідовності

Нехай є деякий скінченний словник токенів V = $\{w_0, w_1,..w_n\}$, та простір цільової змінної (можливі класи) Y. Задача розмітки послідовності полягає в тому, щоб побудувати модель, щоб для довільної послідовність T = $\{t_0, t_1,..t_m\}$, при чому $\forall i = 1, m\ t_i \in V$ оцінити таку величину: $P_i(y=y_j \mid t_0, t_1,..t_m; \Theta)$ - ймовірність, що і-й токен належить до класу y_j за умови послідовності токенів $t_0, t_1,..t_m$; Θ - параметри моделі.

Нехай є деяка послідовність T = $\{t_0, t_1,..t_m\}$, для якої відомі $\{y_0, y_1,..y_m\}$ - реальні значення цільової змінної (мітки класу). Тоді параметри моделі Θ можемо оцінити за допомогою методу максимальної правдоподібності:

$$L = \prod_{i=1}^{m} P_i(y_i \mid t_0, t_1,..t_m; \Theta) \rightarrow max \Leftrightarrow \sum_{i=1}^{m} log(P_i(y_i \mid t_0, t_1,..t_m; \Theta)) \rightarrow max$$

Найбільш поширеним випадком задачі розмітки послідовності є задача розпізнавання іменованих сутностей - цільовими класами є імена людей, назви організацій та географічні назви тощо. (Рисунок 2.1)

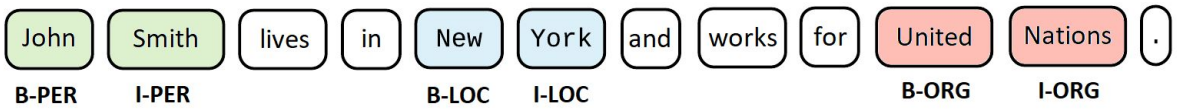


Рисунок 2.1

2.2.3 Задача відкидання хибно-позитивних кандидатів

В силу об’єктивних технічних обмежень, робота з великими послідовностями слів (> 512) є зазвичай неефективною у випадку рекурентних нейронних мереж, зокрема LSTM, та зовсім неможливою у випадку трансформерів кодувальників, таких як BERT. Тому, якщо текстовий документ є великим, його зазвичай розбивають на речення або абзаци перед поданням на вхід до моделі. Але при цьому в кожній розділеній частині може втрачатися контекст і деяка глобальна інформація. Також можливою є ситуація, в якій справжнє цільове словосполучення, що відповідає заданому типу c_k в документі зустрічається лише 1 раз, проте оскільки модель відпрацьовує окремо на реченнях або абзацах, то може втрачатися загальна картина й у декількох абзацах знайтися різні кандидати на релевантну до c_k інформацію й зрештою більшість з них насправді виявляться нерелевантними. Наприклад, розглянемо в якості c_k - “офіційна назва компанії” і будемо розглядати в якості досліджуваних документів деяку звітність чи робочу документацію організації скажімо “The Best Company, LLC”. При чому, очевидним чином, в вхідних документах можуть фігурувати варіанти назви без відповідних корпоративних елементів, що не є релевантним до c_k, і якщо бачити тільки одне речення чи абзац без загального контексту зазвичай неможливо сказати чи є така назва офіційною (релевантною c_k), чи ні.

Саме для унеможливлення вищеописаної ситуації на цьому етапі будемо будувати модель, на вхід якої йдуть токени, розмічені деякою моделлю розмітки послідовностей як такі, що належать до одного з $\{c_1,..c_k\}$ заданих класів в задачі, а також додаткові мета-фактори, які неохідно відбирати й аналізувати вручну відповідно до задачі й предметної області. Задача полягає в тому, щоб прибрати з множини релевантних токенів згідно моделі розмітки

послідовностей такі токени, які з урахуванням загального контексту не є релевантними відповідним c_k класам.

Насправді така задача зводиться до звичайної бінарної класифікації, де X = {(t, p, d)} - простір факторів, що складається з таких трійок: t - токен, p - розподіл ймовірностей для t від деякої моделі розмітки послідовностей, d - вектор додаткових мета-факторів. Зауважимо, що якщо не задані фактори d, то подальша побудова моделі не має сенсу, оскільки реально не додається ніяких додаткових факторів, тому вважатимемо, що фактори d не є тривіальними.
Y = {0, 1} - простір цільової змінної, де 0 - відповідає хибно-позитивному токену, а 1 - істинно-позитивному.

Взагалі кажучи таку задачу можна вирішувати за допомогою різних моделей. Розглянемо модель логістичної регресії.

$$h_\theta(X) = \frac{1}{1 + e^{-\theta^T X}} = \Pr(Y = 1 \mid X; \theta)$$

Оскільки розглядається бінарний випадок: $\Pr(Y = 0 \mid X; \theta) = 1 - h_\theta(X)$

Тоді $\Pr(y \mid X; \theta) = h_\theta(X)^y (1 - h_\theta(X))^{(1-y)}$

Оцінювати параметри моделі будемо за методом максимальної правдоподібності, тому запишемо функцію правдоподібності:

$$\begin{aligned} L(\theta \mid x) &= \Pr(Y \mid X; \theta) \\ &= \prod_i \Pr(y_i \mid x_i; \theta) \\ &= \prod_i h_\theta(x_i)^{y_i} (1 - h_\theta(x_i))^{(1-y_i)} \end{aligned}$$

$$N^{-1} \log L(\theta \mid x) = N^{-1} \sum_{i=1}^{N} \log \Pr(y_i \mid x_i; \theta)$$

Таким чином, отримали $log(L(\Theta|x)) = \sum_{i=1}^{N} (y_i log(\widehat{y_i}) + (1 - y_i) log(1 - \widehat{y_i}))$, де $\widehat{y_i} = h_\Theta(x_i)$

Максимізувати логарифм правдоподібності будемо за допомогою стохастичного градієнтного методу. Для цього наведемо формулу для обчислення градієнту для параметрів моделі:

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial\theta_j}g(\theta^T x) \\
&= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)g(\theta^T x)(1-g(\theta^T x)\frac{\partial}{\partial\theta_j}\theta^T x \\
&= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right)x_j \\
&= (y - h_\theta(x))\,x_j
\end{aligned}$$

2.2.4 Побудова загального підходу до вирішення задачі видобутку текстової інформації

Нехай необхідно робити екстракцію інформації для таких заданих класів $\{c_1,..c_k\}$. Тоді для успішної імплементації підходу необхідно зібрати тренувальну вибірку з відповідної предметної області вигляду $\{(x_i , y_i)\}$,

де $x_i = [t_0, t_1,..t_m]$ - документ (послідовність слів), $y_i = [L_0, L_1,..L_m]$ - розмітка, при чому t_i є V - деякий скінченний словник, L_i є $\{O, c_1,..c_k\}$, де до класу O належатимуть всі токени, що не є релевантнимим жодному з $\{c_1,..c_k\}$. Тоді фінальна модель складатиметься з двох модулів:

- Модель розмітки послідовності, яку тренують безпосередньо на вхідній вибірці $\{(x_i , y_i)\}$. В роботі в якості цього модулю буде розглянуто LSTM модель, та модель на основі кодувальника BERT. Також для змешення мінімально необхідного розміру вибірки слід використовувати методи трансферного навчання, що теж буде розглянуто.

- Модель відкидання хибно-позитивних кандидатів, яка відпрацьовує на результатах роботи попередньої моделі й для тренування потребує людського якісного аналізу факторів. Необхідність і принцип її побудови було детально описано в попередньому пункті. В частині експериментальних досліджень буде розглянуто моделі логістичної регресії та дерева рішень в якості моделі відкидання хибно-позитивних кандидатів. Зауважимо, що потреби в цьому модулі немає при відносно невеликому відсотку хибно-позитивних результатів від попередньої моделі або у разі неможливості використання окремих додаткових факторів на цьому етапі.

Зобразимо схему цього підходу на рисунку 2.2

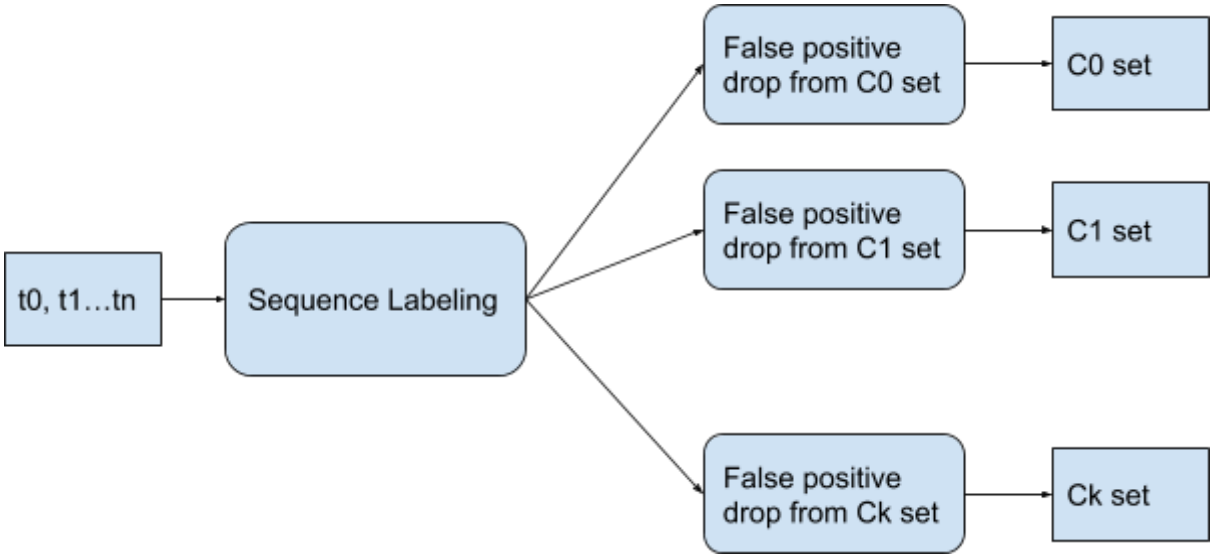
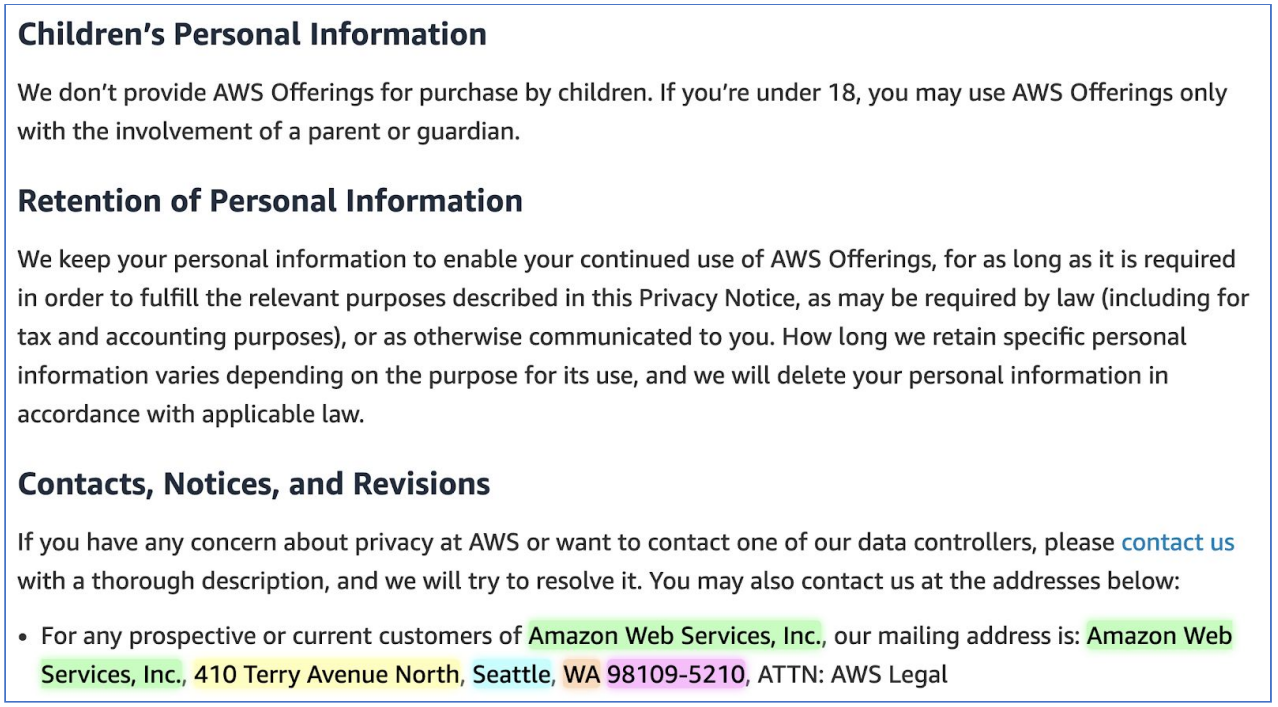


Рисунок 2.2

2.3 Огляд прикладу задачі та тренувальної вибірки

В конкретній задачі, на прикладі якої будемо проводити подальші експерименти маємо в якості основних класів: “офіційна назва організації” та “адреса”, при чому адресу є необхідність розділяти на такі окремі складові як: країна, місто, поштовий індекс, область (штат), адреса 1 (вулиця та номер будинку) та адреса 2 (поверх, номер квартири / апартаментів). Таким чином, маємо в цій задачі 7 класів за якими будемо вести екстракцію з текстових джерел, в якості яких маємо власну вибірку - завантажені тексти з сайтів різних компаніх, для яких є людська анотація для відповідних класів. Всього в цій вибірці маємо 5376 сторінок англійською мовою, здебільшого американських компаній. На рисунку 2.3 зображено приклад розмітки з тренувальної вибірки, а на рисунку 2.4 очікуваний результат роботи алгоритму:



Children’s Personal Information

We don’t provide AWS Offerings for purchase by children. If you’re under 18, you may use AWS Offerings only with the involvement of a parent or guardian.

Retention of Personal Information

We keep your personal information to enable your continued use of AWS Offerings, for as long as it is required in order to fulfill the relevant purposes described in this Privacy Notice, as may be required by law (including for tax and accounting purposes), or as otherwise communicated to you. How long we retain specific personal information varies depending on the purpose for its use, and we will delete your personal information in accordance with applicable law.

Contacts, Notices, and Revisions

If you have any concern about privacy at AWS or want to contact one of our data controllers, please contact us with a thorough description, and we will try to resolve it. You may also contact us at the addresses below:

- For any prospective or current customers of Amazon Web Services, Inc., our mailing address is: Amazon Web Services, Inc., 410 Terry Avenue North, Seattle, WA 98109-5210, ATTN: AWS Legal

Рисунок 2.3

Рисунок 2.4

Наведемо приклад формату вхідних даних у CoNLL-2003 [9] форматі для тренування моделі розмітки послідовності (Рисунок 2.5):

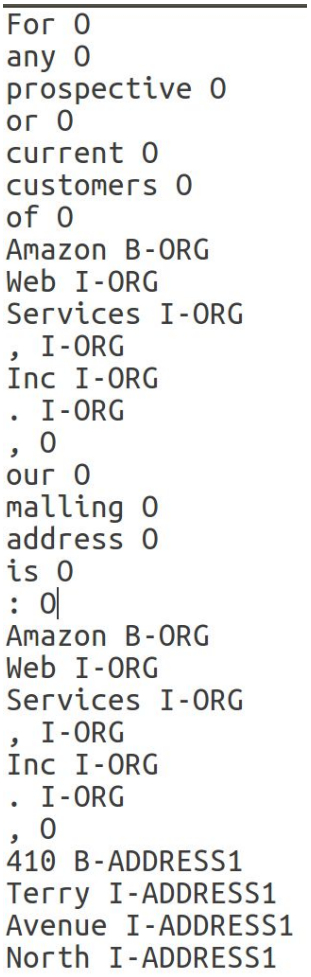


Рисунок 2.5

Зауважимо, що дуже часто цілісні сутності складаються з декількох токенів, але на момент токенізації це невідомо й тому в загальному випадку для класу TYPE вводять B-TYPE (перший токен сутності типу TYPE), та I-TYPE (продовження сутності типу TYPE) [10]. Таку техніку ми будемо використовувати для всіх класів в нашому випадку й на рисунку 2.5 в прикладі розмітки це добре видно.

Наведемо гістограму розподілу кількості токенів в текстах тренувальної вибірки на рисунку 2.6, значення деяких статистик для кількості токенів в таблиці 2.1 та частоти наявності в текстах токенів класів, що нас цікавлять в рамках конкретної задачі в таблиці 2.2

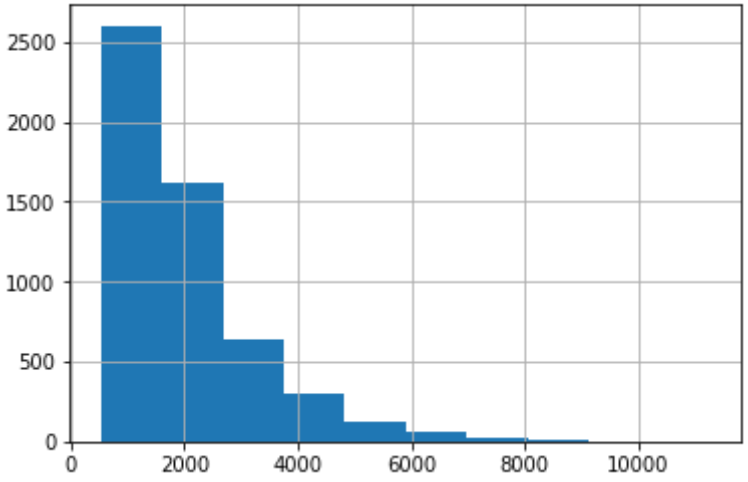


Рисунок 2.6

Таблиця 2.1 - деякі статистики розподілу кількості токенів в документах

Мінімальна довжина	521
Максимальна довжина	11290
Середня довжина	2006.8
Середньокв. відхилення довжини	1257.5
Медіанна довжина	1639

Таблиця 2.2 - розподіл наявності класів в документах

Клас	Частота
ORG	0.78
ADDRESS1	0.51
ADDRESS2	0.12
CITY	0.53
STATE	0.34
ZIP	0.49
COUNTRY	0.10

Також для застосування методів трансферного навчання, зокрема подвійного попереднього тренування було зібрано ще одну вибірку текстів з сайтів кількістю 50000, для якої, на жаль, немає людських анотацій, і ми будемо використовувати її для підходу послідовного трансферного навчання - перенесення на іншу задачу в рамках однієї предметної області. Зауважимо, що значення статистик кількості токенів для цієї вибірки схожі на значення для основної, для якої маємо розмітку.

2.4 Експериментальні дослідження

2.4.1 Метрика

Для оцінки якості будемо використовувати F1 метрику з макро усередненням. $MacroF1 = 2 * \frac{MacroPrecision * MacroRecall}{MacroPrecision + MacroRecall}$

Для кожного i-го класу, розраховується $\text{Precision}_i = \frac{TruePositives_i}{TruePositives_i + FalsePositives_i}$ та $\text{Recall}_i = \frac{TruePositives_i}{TruePositives_i + FalseNegatives_i}$, де TruePositives_i-кількість прикладів, які були класифіковані як такі, що належать до і й вони дійсно належать до і.

FalsePositives_i - кількість прикладів, які були класифіковані як такі, що належать до і й вони до і при цьому до і вони насправді не належать.

FalseNegatives_i - кількість прикладів, які були класифіковані як такі, що не належать до і, хоча при цьому насправді вони до нього належать.

Зобразимо це схематично на рисунку 2.7

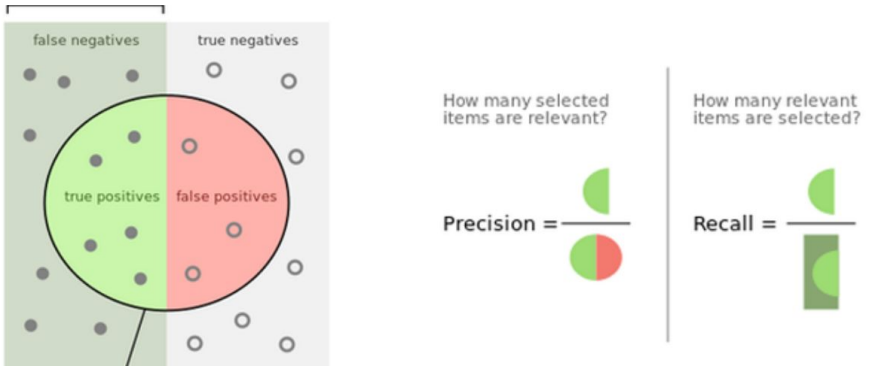


Рисунок 2.7

Тоді, $MacroPrecision = \frac{1}{K}\sum_{i=1}^{K} Precision_i$; $MacroRecall = \frac{1}{K}\sum_{i=1}^{K} Recall_i$, де K - кількість класів.

Таким чином, можемо обрахувати MacroF1.

2.4.2 Валідаційна стратегія

Будемо випадковим чином розбивати нашу початкову вибірку на тренувальну, валідаційну, та тестову частини у відсотковому відношенні 70/15/15. Щоб запобігати ефекту “перенавчання” моделі, після завершення роботи алгоритму тренування, в якості фінальних оцінок параметрів моделі

виберемо такі, що максимізують значення цільової метрики на відкладеній валідаційній підвибірці, але оскільки параметри конструктивно вибирали так, щоб максимізувати значення метрики на валідаційній вибірці, то це значення зазвичай в середньому буде кращим, ніж на будь-якій іншій довільній відкладеній вибірці, тому в якості оцінки якості роботи моделі будемо використовувати значення цільової метрики на тестовій підвибірці, приклади з якої взагалі ніяк не впливали на оцінку параметрів моделі. Схема такого розбиття представлена на рисунку 2.8.

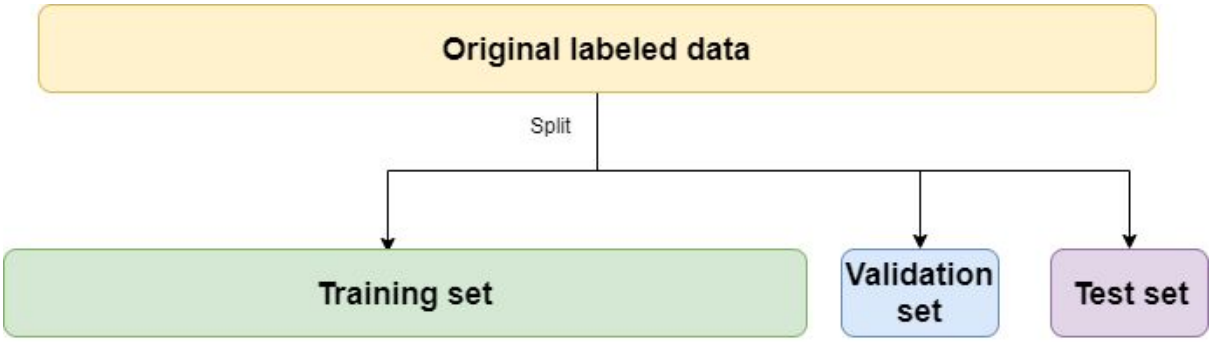


Рисунок 2.8

Також, оскільки в нас як для задачі застосування глибокого навчання відносно не дуже багато даних, то для більш об’єктивної оцінки якості роботи моделі будемо виконувати всю цю процедуру декілька разів (5) для різних випадкових розбиттів, а результати за метрикою будемо усереднювати.

2.4.3 Введення функції “втрат” та особливості чисельної оптимізації

Нагадаємо, що для послідовності $T = \{t_1, t_2,..t_m\}$, для якої відомі $\{y_1, y_2,..y_m\}$ - реальні значення цільової змінної (мітки класу), параметри моделі Θ можемо оцінити за допомогою методу максимальної правдоподібності:

$$L = \prod_{i=1}^{m} P_i(y_i \mid t_0, t_1,..t_m;\ \Theta) \rightarrow max \Leftrightarrow \sum_{i=1}^{m} log(P_i(y_i \mid t_0, t_1,..t_m;\ \Theta)) \rightarrow max$$

Тоді для вибірки різних текстів $D = \{T_1, T_2,..T_n\}$; $T_i = \{t_{i,1}, t_{i,2},..t_{i,mi}\}$, для яких відомі відповідні значення цільової змінної $Y = \{Y_1, Y_2,..Y_n\}$; $Y_i = \{y_{i,1}, y_{i,2},..y_{i,m}\}$, логарифм функції правдоподібності виглядає так:

$\sum_{i=1}^{n} \sum_{j=1}^{mi} log(P_j(y_{i,j} \mid t_{i,1}, t_{i,2},..t_{i,mi};\ \Theta))$, де Θ - параметри моделі,

$P_j(y_{i,j} \mid t_{i,1}, t_{i,2},..t_{i,mi};\ \Theta)$ - ймовірність, що j-й токен послідовності $t_{i,1}, t_{i,2},..t_{i,mi}$ належить до класу $y_{i,j}$ за умови $t_{i,1}, t_{i,2},..t_{i,mi}$ та параметрів Θ. Хочемо побудувати модель, щоб оцінити цю ймовірність.

Нехай $h([t_1, t_2,..t_m;\ \Theta])_i = [p_1, p_2,... p_K]$ - розподіл ймовірностей належності до К класів і-го токена послідовності $[t_1, t_2,..t_m]$; $\sum_{j=1}^{K} p_j = 1$; Для і - го токену відоме значення цільової змінної - його належність до одного з К класів, яку представимо у вигляді одиничного вектору з 1 в позиції, що відповідає індексу відповідного класу: $y_i = [0, 0, \ldots 0, 1, 0, \ldots 0]$, тоді:

$$P_i(y_i \mid t_1, t_2,..t_m;\ \Theta) = \prod_{j=1}^{K} (h([t_1, t_2,..t_m;\ \Theta])_{i,j})^{y_{i,j}}$$, де

$h([t_1, t_2,..t_m;\ \Theta])_{i,j}$ - j-та компонента $[p_1, p_2,... p_K]$, тобто ймовірність належності і-го токену то класу j; $y_{i,j}$ - j-та компонента цільового одиничного вектору.

Тоді $\log(P_i(y_i \mid t_1, t_2,..t_m;\ \Theta)) = \sum_{j=1}^{K} y_{ij} \log(h([t_1, t_2,..t_m;\ \Theta])_{i,j}) =$

$= CE(y_i, h([t_1, t_2,..t_m;\ \Theta])_i)$; CE -так звана функція перехресної ентропії.

Таким чином, логарифмічна функція правдоподібності за всією вибіркою виглядає так (зовнішня сума пробігає по текстам, внутрішня по словах всередині):

$$\sum_{i=1}^{n} \sum_{j=1}^{mi} CE(y_{i,j,}\ h([t_{i,1}, t_{i,2},..t_{i,mi};\ \Theta])_j)$$

Максимізовувати цю функцію можемо методом стохастичного градієнтного спуску. Вигляд власне градієнта залежатиме від вигляду функції h.

В нашому випадку, будемо будувати функцію h на основі раніше згаданих кодувальників (LSTM, BERT). Позначимо функцію кодувальника як E.

E: $\{x_1, x_2,..x_m\} \rightarrow \{y_1, y_2,..y_m\}$ - відображення вхідної векторної послідовності $\{x_1, x_2,..x_m\}$ в результат роботи кодувальника - вихідну векторну послідовність $\{y_1, y_2,..y_m\}$. Щоб для виходу кодувальника y_i оцінити розподіл ймовірностей належності до класів {1, 2, .. K} застосуємо до виходів лінійний оператор W (також належить до параметрів моделі), що діє з простору такої ж розмірності, що й y_i в простір розмірності K, та застосуємо до Wy_i функцію softmax.

$softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$, де z - вектор розмірності K. Зауважимо, що $\sum_{j=1}^{K} softmax(z)_j = 1$.

Тоді $h([t_1, t_2,..t_m ; \Theta])_i = softmax(W(E(t_1, t_2,..t_m; \Theta)_i))$, де Θ - параметри шару кодувальника; W - матриця параметрів класифікаційного шару. Θ та W оцінюються під час процесу тренування моделі - максимізації правдоподібності за тренувальною вибіркою. E - деяки кодувальник (в наших дослідженнях LSTM чи BERT)

2.4.4 Базовий підхід

Базовий підхід полягає в тому, що б виконувати тренування для моделей на нашій вибірці починаючи “з нуля” від випадкової ініціалізації вагів. Зазвичай ваги ініціалізують накшталт $w_{ij} \sim \text{Uniform}(-r, r)$, $r = \frac{1}{\sqrt{dim_{in}+dim_{out}}}$, де dim_{in}, dim_{out} - вхідна та вихідна розмірності матриці W відповідно. Будемо робити випадкову ініціалізацію для всіх модулей моделі: ембеддінг матриця, кодувальник, класифікатор. Оптимізацію виконуватимемо за допомогою модифікації методу стохастичного градієнтного спуску Adam [13], параметри моделі вибираємо такі, які дають максимальний результат на валідаційні підвибірці. В таблицю результатів (Таблиця 2.3) записуємо середнє значення та середньоквадратичне відхилення метрики на тестовій вибірці для 5 різних розбиттів.

Таблиця 2.3

Модель	середнє MacroF1	с.кв. відх. MacroF1
LSTM	0.6770	0.0116
BERT	0.6642	0.0184

Як бачимо результати не є вражаючими оскільки тренувальна вибірка є відносно малою, а моделі дуже “важкими” з великою кількістю параметрів, тому дуже швидко спостерігаємо ефект перенавчання й невисокі результати на тестовій вибірці.

2.4.5 Підхід трансферного навчання: адаптація предметної області та задачі

Цей підхід полягає в тому, що б дотренувати для нашої вибірки та задачі модель, яку вже попередньо тренували на великому обсягу даних для деякої іншої задачі. Для LSTM варіанту візьмемо модель FLAIR [12]: ембеддінги - flair forward-backward ембеддінги [11], кодувальник - з претренованої для задачі розпізнавання іменованих сутностей задачі на датасеті CoNLL03 [9][12] (найпопулярніший бенчмарк для цієї задачі), класифікатор - випадкова ініціалізація. Для BERT варіанту для ембеддінгу і кодувальника візьмемо ваги англомовної моделі з офіційного репозиторію [8]. Там модель була претренована на класичній задачі маскованого мовного моделювання. Оптимізацію виконуватимемо за допомогою модифікації методу стохастичного градієнтного спуску Adam [13], параметри моделі вибираємо такі, які дають максимальний результат на валідаційні підвибірці. В таблицю результатів (Таблиця 2.4) записуємо середнє значення та середньоквадратичне відхилення метрики на тестовій вибірці для 5 різних розбиттів.

Таблиця 2.4

Модель	середнє MacroF1	с.кв. відх. MacroF1
LSTM	0.6941	0.0072
BERT	0.7522	0.0089

Як бачимо результати є значно кращими, ніж в попередньому варіанті та дисперсія значення метрики по різних робиттях є також меншою.

2.4.6 Підхід трансферного навчання: адаптація задачі

Цей підхід полягає в тому, що б зробити попереднє тренування моделей з попереднього підходу спочатку на даних з релевантної предметної області (можливо для іншої задачі), і тільки після того тренувати для нашої вибірки і задачі. Для LSTM варіанту візьмемо модель FLAIR: ембеддінги - flair forward-backward ембеддінги, кодувальник - з претренованої для задачі розпізнавання іменованих сутностей задачі на датасеті CoNLL (найпопулярніший бенчмарк для цієї задачі), класифікатор - випадкова ініціалізація. Для BERT варіанту для ембеддінгу і кодувальника візьмемо ваги англомовної моделі з офіційного репозиторію. Там модель була претренована на класичній задачі маскованого мовного моделювання.

Нагадаємо, що в пункті 2.3 було також описано вибірку нерозмічених сайтів. Будемо використовувати ці дані в якості даних з релевантної предметної області. Задачу попереднього тренування сформуємо таким чином: візьмемо модель FLAIR для задачі розпізнованих сутностей, натреновану на датасеті CoNLL і будемо розмічати за допомогою неї в нашій проміжній вибірці тільки теги ORG та LOC. В CoNLL ORG відповідає будь яким назвам організацій (не обов’язково офіційним), LOC - агрегований тег для будь-яких географічних об’єктів чи адрес. Натренуємо моделі BERT та LSTM для цієї проміжної задачі. Потім замінимо в них останній класифікаційний шар на шар відповідної для нашої задачі розмірності з випадковою ініціалізацією вагів. Оптимізацію виконуватимемо за допомогою модифікації методу стохастичного градієнтного спуску Adam, параметри моделі вибираємо такі, які дають максимальний результат на валідаційні підвибірці. В таблицю результатів (Таблиця 2.5)

записуємо середнє значення та середньоквадратичне відхилення метрики на тестовій вибірці для 5 різних розбиттів.

Таблиця 2.5

Модель	середнє MacroF1	с.кв. відх. MacroF1
LSTM	0.7549	0.0048
BERT	0.8204	0.0032

Як бачимо результати є значно кращими, ніж в попередньому варіанті та дисперсія значення метрики по різних робиттях є також меншою.

2.4.7 Відкидання хибно-позитивних кандидатів

В цьому пункті розглядатиметься застосування простої класифікаційної моделі, описаної в 2.2.3 для відкидання хибно-позитивних кандидатів, отриманих в результаті роботи моделей, описаних в попередніх пунктах. Будемо використовувати цю техніку для ORG класу. Сформуємо нову вибірку таким чином: розіб’ємо всю вибірку випадковим чином на 5 частин, далі для кожної частини будемо навчати модель розмітки послідовності на інших чотирьох і брати результати оцінок ймовірностей класу ORG для токенів для нашої п’ятої частини. Тепер із кандидатів на ORG згідно оцінок моделі сформуємо вибірку, де для кожного кандидата задамо значення цільової змінної так: 1 - якщо кандидат від моделі дійсно належить до ORG (істинно-позитивний кандидат), і 0 - якщо не належить (хибно-позитивний кандидат). Оскільки працюємо з конкретною задачею, після деякого попереднього аналізу можемо сформувати фактори, на яких будемо тренувати модель логістичної регресії. Також створимо функцію схожості між двома іменами таким чином: будемо

рахувати різниці в кількості уніграм, біграм та триграм між двома іменами після застосування нормалізації - зведення імені до нижнього регістру, вилучення корпоративних елементів а також всіх небуквених символів і за можливістю застосування лемматизації та стемізації.

Таким чином, маємо такі фактори:

- Схожість нормалізованого кандидата і нетлоку посилання веб-сторінки
- Середня схожість нормалізованого кандидата на інших нормалізованих кандидатів з даної сторінки
- Кількість символів
- Наявність корпоративного елемента (бінарний фактор)
- Чи містить даний кандидат найбільшу кількість символів серед усіх інших на даній сторінці (бінарний фактор)
- Чи має цей кандидат найбільшу середню схожість серед усіх іних на даній сторінці (бінарний фактор)

Перед безпосередньо моделюванням для небінарних факторів застосовуємо процедуру стандартизації - віднімаємо середнє і ділимо на середньоквадратичне відхилення за вибіркою. Отримані результати після моделювання запишемо в таблицю 2.6 (BERT кращий результат без відкидання хибно-позитивних ORG кандидатів, BERT + FP drop - з відкиданням).

Таблиця 2.6 - результати з застосуванням відкидання хибно-позитивних кандидатів

Модель	F1 для ORG класу	середнє MacroF1	с.кв. відх. MacroF1
BERT	0.6502	0.8204	0.0032
BERT + FP drop	0.8423	0.8489	0.0030

2.5 Висновки до розділу 2

В цьому розділі описано формалізацію абстрактної задачі та загальний підхід до її вирішення через вирішення задачі розмітки послідовності та відкидання хибно-позитивних кандидатів. Було розглянуто приклад конкретної задачі та описано вибірки даних, які використовувалися для її вирішення. Для експериментальних досліджень було описано метрику та стратегію оцінки якості результатів роботи моделей, показано шлях виведення оцінок параметрів моделей за допомогою методу максимальної правдоподібності. Власне експерименти було проведено для двох типів моделей: з LSTM кодувальником та BERT кодувальником для трьох підходів: базового, трансферного навчання з адаптацією предметної області й задачі; трансферного навчання з адаптацією задачі; та трансферного навчання з адаптацією задачі з відкиданням хибно-позитивних кандидатів (Таблиця 2.7). За допомогою останнього підходу отримано найкращий результат. Для базового підходу LSTM виявилася кращою ніж BERT. Це пов’язано з тим, що чим складнішою є модель, тим сильніше спостерігається ефект перенавчання при тренуванні на відносно невеликих вибірках. В трансферних підходах, доволі очевидно, що кращою виявилася модель з BERT кодувальником, оскільки там обсяг даних для попереднього тренування є дуже великим і відповідно більш складна модель може точніше розуміти різні окремі нюанси та аспекти природної мови.

Таблиця 2.7 - порівняльна таблиця для різних підходів

Підхід	Модель	середнє MacroF1	с.кв. відх. MacroF1
базовий	LSTM	0.6770	0.0116
адаптація домену і задачі	BERT	0.7522	0.0089
адаптація задачі	BERT	0.8204	0.0032
адаптація задачі + відкидання хибно-позитивних	BERT	0.8489	0.0030

РОЗДІЛ 3 АРХІТЕКТУРА ПРОГРАМНОГО ПРОДУКТУ

3.1 Вступ

В цьому розділі буде розглянуто особливості програмної реалізації системи, яку було розроблено в рамках даної дипломної роботи, описано технічні вимоги до системи, контракти прикладного програмного інтерфейсу та наведено різні приклади розгортання системи. Також буде описано різноманітні можливості та функції програмних бібліотек і фреймворків, які були використані для розробки даної системи. Кінцевий програмний продукт представляє собою систему для видобутку інформації з неструктурованих текстових джерел реалізовану як веб-сервіс з описаним прикладним програмним інтерфейсом. Розроблений сервіс складається з окремих модулів - мікросервісів: управляючий сервіс, кроулер (для видобутку тексту з веб-сайтів) та сервіс з моделлю машинного навчання.

Для розробки було використано мову програмування Python та різні додаткові бібліотеки та фреймворки, зокрема Pytorch для імплементації всіх вищеописаних архітектур нейронних мереж. Для розгортання було використано хмарний сервіс AWS. Також для контейнеризації й масштабування сервісу використано Docker і Kubernetes відповідно.

В розділі також буде розглянуто приклади розгортання системи на віртуальній машині та в kubernetes кластері та наведено метрики роботи власне сервісу, зокрема середня затримка, кількість запитів на секунду та інші.

3.2 Основні вимоги до сервісу

1. Сервіс повинен видобувати необхідну інформацію як з довільних текстів, так і з веб-сайтів, з яких це законно дозволено робити.
2. Сервіс має реалізовувати два варіанти роботи: працювати в режимі реального часу по схемі запит-відповідь для кожного окремого документу, а також реалізовувати пакетну обробку (batch transformation) для масиву документів по схемі зчитати з бази даних документи - записати в базу даних результат для всіх документів, можливо, за тривалий період часу.
3. Мікросервіс моделі машинного навчання повинен мати можливість розгортання як на машинах тільки з процесором (CPU), так і на машинах, з графічними обчислювальними пристроями (GPU) з метою пришвидшення роботи системи.
4. Сервіс має підтримувати механізм автоматичного горизонтального масштабування з метою загального пришвидшення роботи системи.

3.3 Опис контрактів прикладного програмного інтерфейсу

Спочатку наведемо опис основних типів LabeledORG (рисунок 3.1) та LabeledADDR (рисунок 3.2), що відповідають розміченим офіційному імені компанї, та адресі відповідно.

```
LabeledORG
{
    "text": string,
    "rank": float
}
```

Рисунок 3.1 - Опис типу LabeledORG

```
LabeledADDR
{
    "ADDR1": string,
    "ADDR2": string,
    "CITY": string,
    "STATE": string,
    "COUNTRY": string,
    "ZIP": string,
    "rank": float
}
```

Рисунок 3.2 - Опис типу LabeledADDR

Зауважимо, що в LabeledORG полю “text” відповідає власне розмічена частина вхідного тексту, що відповідає офіційній назві компанії, а полю “rank” відповідає ймовірність належності до класу ORG згідно моделі машинного навчання. В LabeledADDR усім полям, окрім rank відповідають відповідні компоненти адреси описані в пункті 2.3, полю rank відповідає агрегований ранг для всієї адреси отриманий від моделі машинного навчання.

Відповідно до основних вимог, а саме про можливість опрацювання окремих текстів, веб-сайтів, та таблиць, що містять багато текстів або посилань на сайти, прикладний програмний інтерфейс має такі методи:

- api/process_text - для опрацювання простого тексту (рисунок 3.3).
- api/process_url - для опрацювання веб-сайту за його посиланням (рисунок 3.4).
- api/process_table - для опрацювання таблиці (PostgreSQL), що містить тексти або посилання на сайти (рисунок 3.5).

Зауважимо, що в api/process_table в запиті поле type має містити “text”, або “url”, що характеризує колонку, яка буде опрацьована в таблиці, “text” - для текстів, “url” - для посилань на сайти відповідно.

```
api/process_text:

request:
{
    "text": string
}

response:
{
    "ORG": [LabeledORG],
    "ADDRESS": [LabeledADDR]
}
```

Рисунок 3.3 - опис контракту для api/process_text

```
api/process_url:

request:
{
    "url": string
}

response:
{
    "ORG": [LabeledORG],
    "ADDRESS": [LabeledADDR]
}
```

Рисунок 3.4 - опис контракту для api/process_url

```
api/process_table:

request:
{
    "input_table": string,
    "output_table": string,
    "type": string
}

response:
{
    "status": string
}
```

Рисунок 3.5 - опис контракту для api/process_table

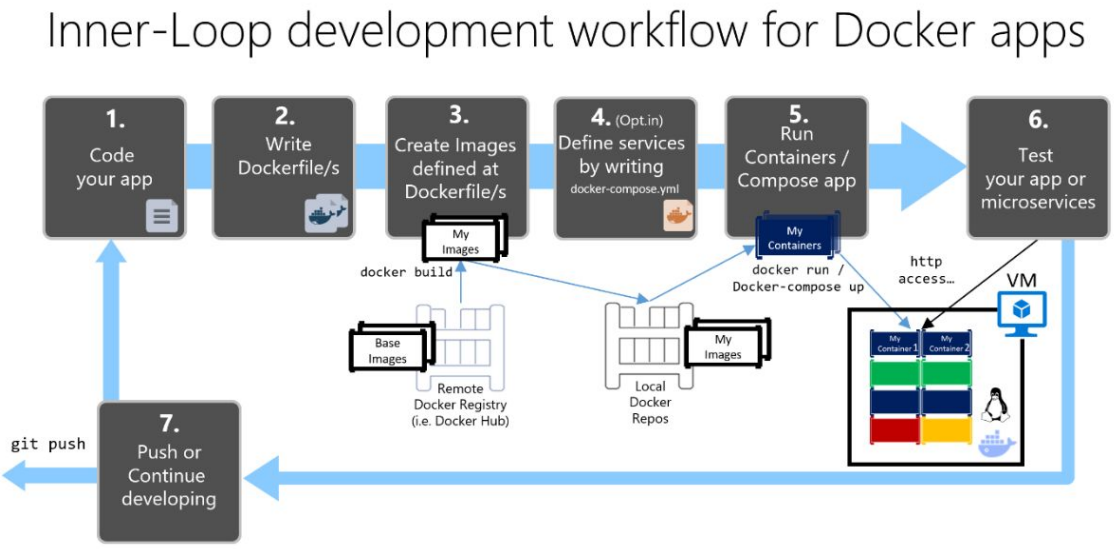
3.4 Вибір мови програмування та опис основних технологій

В якості основної мови програмування було обрано Python, тому що синтаксис цієї мови є доволі простим і інтуїтивним, а також для неї існує велика кількість стандартних та сторонніх бібліотек та фреймворків, зокрема для машинного навчання та розробки веб-сервісів. Python є інтерпретованою мовою, а тому не потрібно витрачати час на компіляцію перед кожним запуском, що суттєво пришвидшує дослідження на етапі експериментів з даними та різними моделями. Оскільки python є кросплатформенним, імплеметовані програмні продукти зазвичай можна використовувати на машинах під управлінням різних операційних систем, зокрема Linux, MacOS, Windows.

Для проведення попередніх експериментів з даними та моделями було використано інтерактивну веб-оболонку Jupyter Lab, що дозволяє створювати окремі комірки програмного коду, які можуть бути виконані інтерпретатором в довільно заданому користувачем порядку. Основними засобами були math, numpy - бібліотеки для обислень та лінійної алгебри, matplotlib для візуалізації, pytorch - бібліотека машинного та глибокого навчання, що реалізує динамічні обчислювальні графи, flair - фреймворк над pytorch для моделей обробки природної мови.

В якості фреймворку для створення веб-сервісу використовується Flask, оскільки цей фреймворк є мінімалістичним й інтуїтивно зрозумілим для освоєння, а також активно підтримується громадою python веб-розробників.

Для контейнеризації сервісів використовується Docker, який дозволяє “запакувати” сервіс із усіма залежностями в контейнер, який може бути перенесений на будь-яку Linux систему. Схема процесу розробки програмних продуктів з використанням Docker зображена на рисунку 3.6.



Рисуок 3.6 - Схема розробки з використанням Docker

Для горизонтального масштабування сервісу з моделями використовуємо Cortex - платформу для автоматичного розгортання сервісів машинного навчання в хмарі AWS, що реалізує зручну для користування модифікацію над Kubernetes. Архітектуру Kubernetes зобразимо на рисунку 3.7.

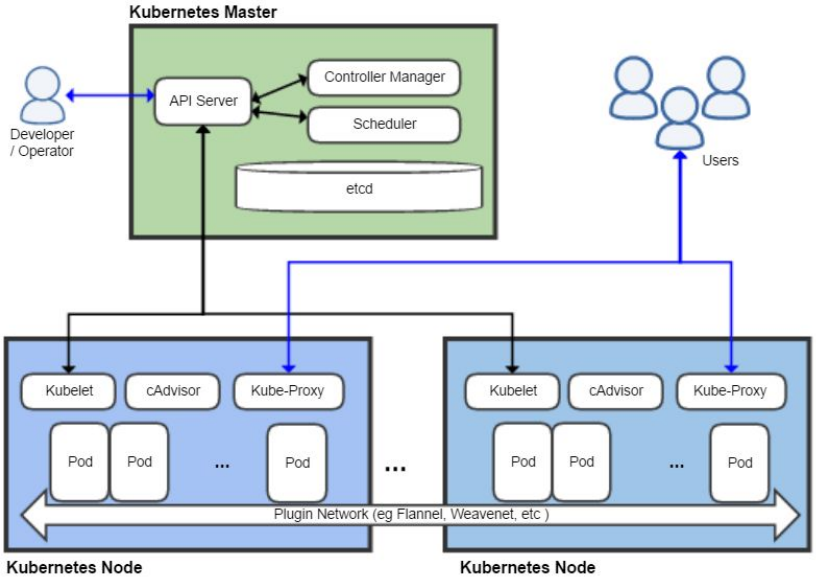


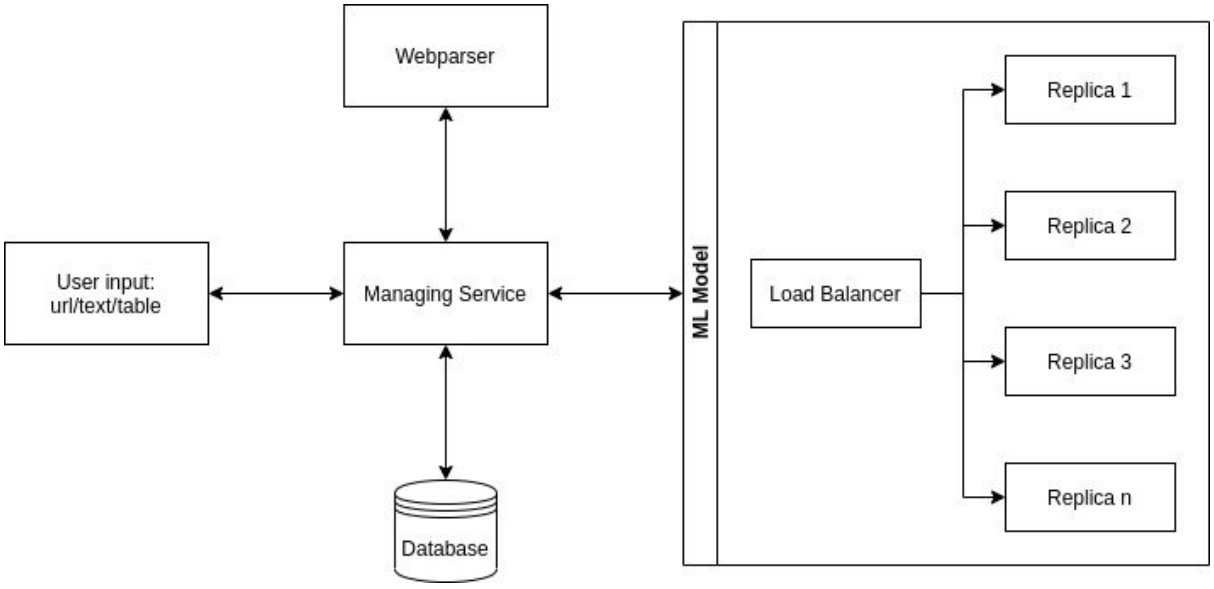
Рисунок 3.7 - Архітектура Kubernetes

В Kubernetes основними є такі поняття:

- Вузол (Node) - окрема реальна або віртуальна машина, в якій розгортаються репліки програмного застосунку.
- Стручок (Pod) - представляє собою базову одиницю для запуску та управління програмним застосунком, що складається з одного або декількох контейнерів.
- Контролер (Controller) - представляє собою процес, що керує станом кластера, зокрема контролер реплікації (Replication Controller) забезпечує масштабування шляхом запуску заданої кількості копій стручка (репліки) в кластері. За допомогою нього також виконується старт нових реплік, якщо старі виходять з ладу.

3.5 Аналіз схеми роботи та архітектури системи

Наведемо загальну схему взаємодії мікросервісів на рисунку 3.8.



Рисунку 3.8 - загальна архітектура системи

Зауважимо, що управляючий сервіс (Managing Service) представляє собою веб-сервіс, який реалізує необхідну поведінку системи в залежності від вибраного методу api/process_text, api/process_url, api/process_table. Webparser - сервіс для екстракції тексту з веб-сторінок, який завантажує сторінку та дістає з неї текстові блоки за можливістю без випадаючих меню та реклам за допомогою сторонньої бібліотеки boilerpipe. Також зауважимо, що власне сама по собі задача розпізнавання релевантного/рекламного тексту та блоків коду на веб-сторінках є окремою задачею і не розгядається в рамках даної роботи, а тому використовуватимемо стандартне рішення. Database - база даних під управлінням PostgreSQL. ML Model - сервіс з репліками моделі машинного навчання під управлінням балансувальника навантаження. Наведемо алгоритм роботи системи на рисунку 3.9.

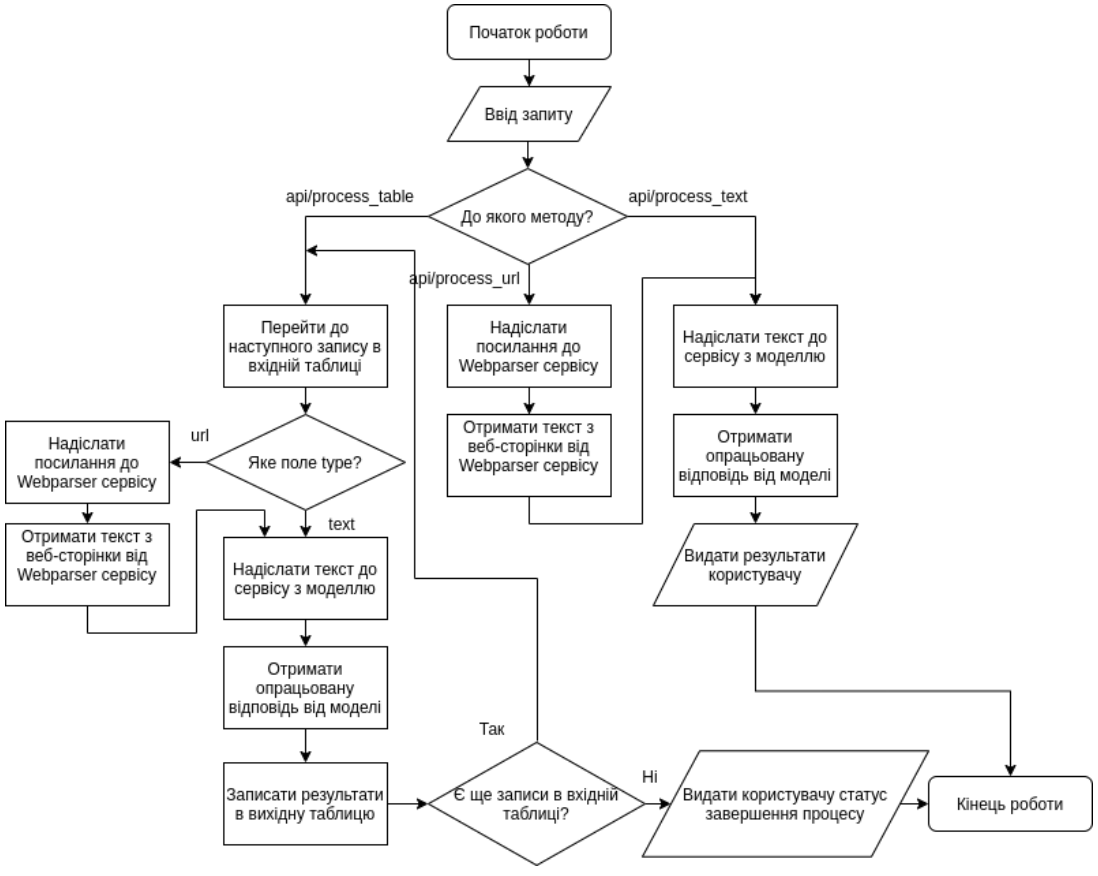


Рисунок 3.9 - Алгоритм роботи системи

3.6 Приклади розгортання системи

Систему будемо розгортати в хмарі AWS, запропонуємо два варіанти розгортання: на віртуальній машині (аналогічним чином можна розгортати й на звичайній фізичній машині) та в кластері. Також для всіх прикладів будемо виконувати тестування навантаження розгорнутої системи за допомогою wrk.

3.6.1 Розгортання системи на віртуальній машині

Спочатку зробимо контейнеризацію всіх трьох сервісів: управляючого, кроулера та моделі з використанням Docker. Зазначимо, що в якості веб-фреймворку будемо використовувати flask, а в якості http веб-серверу використовуватимемо Gunicorn, який обробляє запити за допомогою worker-процесів. Для сервісу з моделлю будемо використовувати стільки worker-процесів, скільки є ядер на машині з метою паралелізації роботи моделі. Зауважимо, що у випадку з моделями машинного навчання маємо справу з процесами, які є обчислювально складними й немає сенсу створювати worker-процеси у кількості, що перевищує кількість ядер машини. Зрештою розгортаємо три сервіси на трьох окремих віртуальних машинах c5.2xlarge на AWS, що мають 8 ядер та 16 Гб оперативної пам’яті. Результати тестування на навантаження обробки сторінок веб-сайтів за допомогою api/process_url наведемо на рисунку 3.10. Серед ключових показників кількість обробки запитів на секунду 0.16 запит./сек., та середній час очікування для одного запиту 39.21 сек. Такі показники не є вражаючими й достатніми для використання в продуктивних системах, але вони легко пояснюються тим, що робота моделі є обчислювально дуже непростим процесом і дані віртуальні

машини не містять графічного прискорювача, а отже, маємо тривалий час обробки запитів. Зазначимо, що завантаження й видобуток тексту з веб-сторінок іноді може бути процесом тривалим, проте обчилювально легким, й основний час там йде здебільшого на очікування відповіді від сайту.

```
Running 10m test @ http://t7347htg543b4a8493062
  40 threads and 40 connections
  Thread Stats  Avg     Stdev    Max  +/- Stdev
    Latency   39.21s   53.45s   2.23m   86.83%
    Req/Sec    0.04     0.12    0.32    91.61%
  Latency Distribution
     50%   8.17s
     75%  42.96s
     90%   1.58m
     99%   2.22m
  97 requests in 10.00m, 93.65KB read
  Socket errors: connect 0, read 0, write 0, timeout 9
  Non-2xx or 3xx responses: 1
Requests/sec:     0.16
Transfer/sec:    159.83B
```

Рисунок 3.10 - WRK тестування на навантаження сервісу на віртуальній машині

3.6.2 Розгортання системи в кластері

З метою пришвидшення роботи сервісу та покращення показника кількості оброблених запитів на секунду будемо розгортати сервіс з моделлю в Kubernetes кластері за допомогою платформи Cortex. При створенні кластеру можемо задавати мінімальну та максимальну бажану кількість віртуальних машин в кластері та максимальну кількість реплік сервісу на одну віртуальну машину. За допомогою механізму автоматичного масштабування відбуватиметься підтримка такої кількості віртуальних машин, яка відповідатиме необхідній кількості реплік сервісу в залежності від навантаження. Тобто, якщо навантаження є більшим, ніж витримує поточна кількість реплік, в кластері додаються нові віртуальні машини, на яких

розгортаються додаткові репліки; якщо ж навантаження є меншим, непотрібні репліки й відповідні віртуальні машини прибираються з кластеру. Управляючий сервіс та кроулер будемо розгортати так само, як і в попередньому випадку. Результати тестування на навантаження обробки сторінок веб-сайтів за допомогою api/process_url в кластері машин c5.2xlarge, що мають 8 ядер та 16 Гб оперативної пам'яті наведемо на рисунку 3.11, а в кластері машин p2.xlarge, що мають графічний прискорювач NVIDIA K80, 4 ядра та 61 Гб оперативної пам'яті наведемо на рисунку 3.12.

```
Running 10m test @ http://a289b43a772bc11ea8493(
  40 threads and 40 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    19.68s    23.22s    1.46m    80.90%
    Req/Sec     0.27      0.81     5.00     92.62%
  Latency Distribution
     50%    6.98s
     75%   34.79s
     90%    0.99m
     99%    1.38m
  488 requests in 10.00m, 199.98KB read
  Socket errors: connect 0, read 0, write 0, timeout 43
  Non-2xx or 3xx responses: 3
Requests/sec:      0.81
Transfer/sec:     341.24B
```

Рисунок 3.11 - WRK тестування на навантаження сервісу в кластері c5.2xlarge

```
Running 30m test @ http://a35199bad72d211eab5620
  40 threads and 40 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     8.65s    10.56s    1.20m    85.21%
    Req/Sec     1.00      1.71    10.00     83.94%
  Latency Distribution
     50%    4.10s
     75%   12.92s
     90%   23.70s
     99%   45.31s
  13709 requests in 30.00m, 6.22MB read
Requests/sec:      7.62
Transfer/sec:      3.54KB
```

Рисунок 3.12 - WRK тестування на навантаження сервісу в кластері p2.xlarge

Очевидним чином, отримали суттєве покращення ключових метрик середньої затримки та кількості запитів на секунду в порівнянні з попереднім варіантом. При чому кількість запитів на секунду для кластеру з p2.xlarge є майже в 10 раз більшою ніж для кластеру з c5.2xlarge. Це легко пояснюється надзвичайною потужністю та ефективністю використання графічних прискорювачів для задач роботи з глибокими нейронними мережами. Зауважимо, що в усіх кластерах використовували не більше ніж 10 віртуальних машин.

3.7 Приклади роботи системи та якісний аналіз отриманих результатів

Покажемо як працює система для довільних сайтів, які не увійшли в тренувальну вибірку. Для цього будемо використовувати Jupyter Lab та бібліотеку для http запитів requests. Перший приклад - сайт https://foxnebraska.com/station/terms?s=18991. Витяги з сайту наведемо на рисунках 3.13 та 3.14.

Thank you for visiting this website, which is operated by an Affiliate of Sinclair Television Group, Inc. ("Sinclair"). This site is one of a network of ad-supported sites operated by Affiliates of Sinclair each of which also operates a local television station (each a "Sinclair Affiliate Site" and, collectively, the "Sinclair Network of Sites"). Each Sinclair Affiliate Site has adopted this privacy statement to the extent applicable. "Affiliate" means a company controlling, controlled by or under common control with another company, or a company which shares common management.

Рисунок 3.13

Notice of Copyright Infringement. If you are an owner of intellectual property who believes your intellectual property has been improperly posted or distributed via this Service, please notify us through our feedback procedure or by sending a notice by U.S. Mail to Sinclair Broadcast Group, Inc., 10706 Beaver Dam Road, Hunt Valley, MD 21030 Attn: Legal Department. Your notice to us must include the following information: (1) a physical or electronic signature of a person authorized to act on behalf of the owner of the copyrighted work allegedly infringed; (2) a description of the copyrighted work or works that allegedly have been infringed; (3) a description of where on the Service the allegedly infringing material appears that will allow us to locate the material; (4) a statement by you that you have a good faith belief that the allegedly infringing use has not been authorized by the copyright owner, its agent, or the law; and (5) a statement by you that the information in your notice is accurate, and under penalty of perjury, that you are authorized to act on behalf of the owner of the copyrighted work that has allegedly been infringed.

Рисунок 3.14

Результати роботи системи покажемо на рисунку 3.15. Як бачимо, система змогла дістати 2 офіційних імені компаній та одну адресу.

```
[1]: import json
     import requests

     service_url = 'http://a35199bad72d211eab5620ee4f799c0f-1681897105.us-west-2.elb.amazonaws.com/api/process_url'
     resp = requests.post(service_url, json={'url': 'http://www.foxnebraska.com/global/story.asp?s=18991'}).text
     resp = json.loads(resp)
     resp
```

```
[1]: {'ORG': [{'text': 'Sinclair Television Group, Inc.', 'rank': 0.930471},
       {'text': 'Sinclair Broadcast Group, Inc.', 'rank': 0.610294}],
      'ADDRESS': [{'ADDR1': '10706 Beaver Dam Road',
        'ADDR2': None,
        'CITY': 'Hunt Valley',
        'STATE': 'MD',
        'ZIP': '21030',
        'COUNTRY': None,
        'rank': 0.851902}]}
```

Рисунок 3.15

В другому прикладі покажемо роботи системи з текстом (рисунок 3.16)

```
[1]: import json
     import requests

     service_url = 'http://a35199bad72d211eab5620ee4f799c0f-1681897105.us-west-2.elb.amazonaws.com/api/process_text'
     sample_text = """AERORATER (\"AERORATER\") assists aviation professionals research, rate, and review a variety of
     service contractors and aviation related professionals (collectively, “Service Providers”).
     The following Terms of Use outline your obligations when using the AERORATER websites, mobile applications, and services.
     If you believe that AERORATER has not adhered to this Privacy Policy you may contact us online or write to us at
     the following address: \n AERORATER LLC. \n 1601 N. Sepulveda Blvd. Suite 702 \n Manhattan Beach,
     California 90266 \n United States of America"""
     resp = requests.post(service_url, json={'text': sample_text}).text
     resp = json.loads(resp)
     resp
```

```
[1]: {'ORG': [{'text': 'AERORATER LLC', 'rank': 0.964912}],
      'ADDRESS': [{'ADDR1': '1601 N. Sepulveda Blvd.',
        'ADDR2': 'Suite 702',
        'CITY': 'Manhattan Beach',
        'STATE': 'California',
        'ZIP': '90266',
        'COUNTRY': 'United States of America',
        'rank': 0.951083}]}
```

Рисунок 3.16

Розглянемо також роботу на прикладі сайту НТУУ “КПІ ім. Сікорського” https://kpi.ua/en (рисунок 3.17).

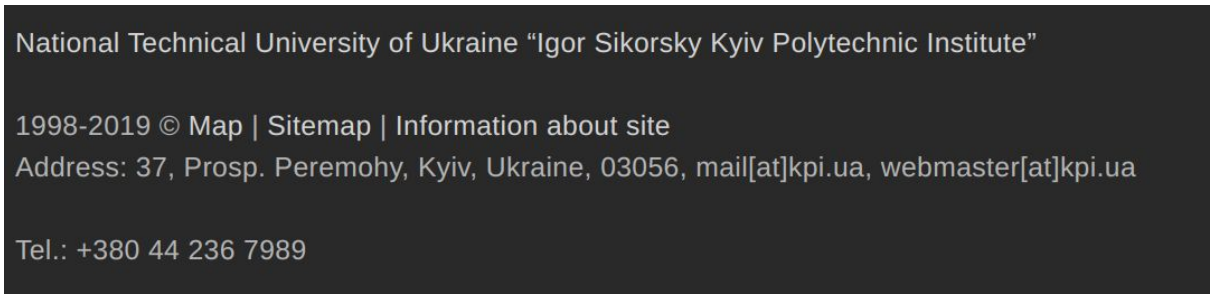


Рисунок 3.17

Результат роботи системи покажемо на рисунку 3.18. На жаль, в даному випадку система спрацювала неідеально, а саме: видала 'Kyiv Polytechnic Institute' в якості офіційного імені установи замість National Technical University

of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, а також пропустила ADDR1, а саме 37, Prosp. Peremohy.

```
[1]: import json
     import requests

     service_url = 'http://a35199bad72d211eab5620ee4f799c0f-1681897105.us-west-2.elb.amazonaws.com/api/process_url'
     resp = requests.post(service_url, json={'url': 'https://kpi.ua/en'}).text
     resp = json.loads(resp)
     resp
```

```
[1]: {'ORG': [{'text': 'Kyiv Polytechnic Institute', 'rank': 0.729477}],
      'ADDRESS': [{'ADDR1': None,
        'ADDR2': None,
        'CITY': 'Kyiv',
        'STATE': None,
        'ZIP': '03056',
        'COUNTRY': 'Ukraine',
        'rank': 0.751284}]}
```

Рисунок 3.18

Отже, отримані результати на цих та інших прикладах свідчать про те, що система добре працює на текстових даних, що лексично схожі на ті, які були в тренувальній вибірці, зокрема комерційні американські компанії, проте в загальному випадку система може припускатися помилок і неточностей, що можна виправити шляхом розширення тренувальної вибірки даними з інших предметних областей.

3.8 Висновки до розділу 3

В цьому розділі розглянули особливості архітектури та імплементації системи та описали технології, які були використані для створення та розгортання системи. Фінальна версія представляє собою веб систему мікросервісної архітектури з управляючим сервісом, кроулером та сервісом моделі машинного навчання. В розділі було запропоновано варіанти розгортання сервісу на віртуальній машині та в кластері. Зазначимо, що всі варіанти відповідають функціональним вимогам, проте варіанти з розгортанням на віртуальній машині та в кластері на машинах без графічного прискорювача не підійдуть з технічної точки зору для більшості випадків використання в

продуктивних системах за такими метриками як середня затримка та середня кількість обробки запитів на секунду.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ ПРИ ПОБУДОВІ СИСТЕМИ ВИДОБУТКУ ІНФОРМАЦІЇ З ТЕКСТОВИХ ДЖЕРЕЛ

4.1 Постановка задачі

Проводиться оцінка основних характеристик результатів та їх собівартість. Для отримання результатів використовувалась мова Python. Середовище розробки - PyCharm та Jupyter Lab. Наведемо аналіз різних підходів до створення та розгортання системи видобутку інформації з текстових джерел на основі LSTM та BERT кодувальників.

4.2 Обґрунтування функцій дослідження

Основні функції:

1. F_1 - вибір стратегії тренування мережі : а) тренування від випадкової ініціалізації; б) тренування від ініціалізації попередньо натренованих моделей і подальше перенесення задачі та предментої област; в) попереднє тренування на схожій задачі в тій же предметній області й подальше перенесення задачі.
2. F_2 - вибір архітектури кодувальника: а) LSTM; б) BERT;
3. F_3 - вибір схеми розгорвання системи: а) розгортання на віртуальній машині; б) розгортання в Kubernetes кластері;

На рисунку 4.1 зобразимо відповідну морфологічну карту системи.

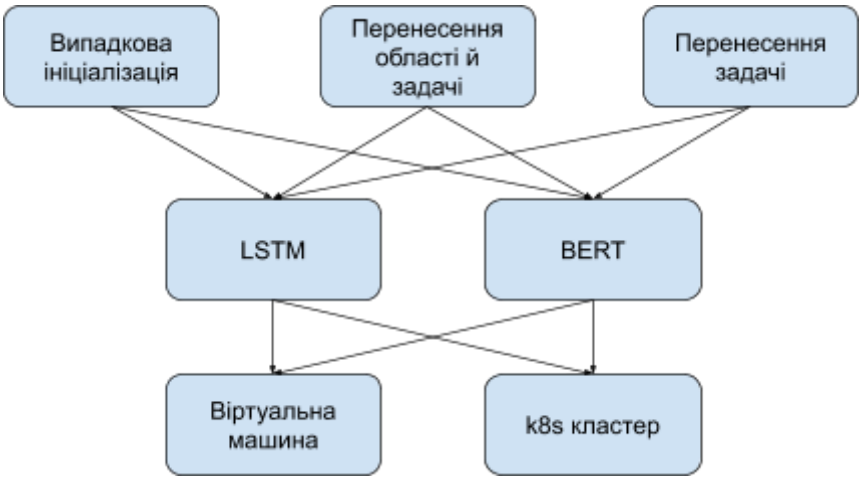


Рисунок 4.1

Позитивно-негативна матриця варіантів основних функцій (табл. 4.1):

Таблиця 4.1

Основні функції	Варіанти реалізацій	Переваги	Недоліки
F_1	А	Найшвидший варіант	Низька якість отриманої моделі за метрикою f1
F_1	Б	Така ж швидкість як і в А. Суттєве покращення метрики f1	Вимагає додаткової даних, а саме попередньо натренованих моделей.
F_1	В	Дає найкращі показники за f1 серед усіх	Потребує додаткови даних та часу на додаткове тренування
F_2	А	Швидша при інференсі	Низька якість за метрикою
F_2	Б	Найсучасніша архітектура кодувальника, дає високі результати за метриками	Потребує значної кількості обчислювальних ресурсів.

F_3	А	Простий в реалізації	Неможливо горизонтально масштабувати розгорнуту систему
F_3	Б	Дозволяє автоматичне горизонтальне масштабування розгорнутої системи	Технічно складний в реалізації потребує спеціальних знань в галузі хмарних технологій

На основі порівняльного аналізу варіантів реалізації основних функцій по їх перевагам та недолікам можна виключити варіанти F_1 А, F_1 Б, F_2 А, тоді варіанти, які залишилися:

F_1 В -> F_2 Б -> F_3 А

F_1 В -> F_2 Б -> F_3 Б

Для оцінювання описаних функцій запропонуємо систему параметрів.

Для характеризації досліджень пропонуємо такі параметри:

X_1 — складність освоєння технологічного стеку;

X_2 — середня кількість запитів на секунду, яку здатна витримати система;

X_3 — час на розгортання системи з нуля в хмарному середовищі;

X_4 — середня вартість утримання системи протягом години;

F_1 застосовує параметри X_1 та X_3

F_2 застосовує параметр X_1

F_3 застосовує параметри X_1, X_2, X_3, X_4

Гірші, середні та кращі показники параметрів вибираються на основі вимог замовника та умов перебігу дослідження, їх наведено у таблиці 4.2.

Таблиця 4.2

Умовні позначення	Одиниці виміру	Гірші	Середні	Кращі
X_1	оцінка	1	5	10
X_2	к-ть запит/сек	0.1	1	10
X_3	Год.	168	40	6
X_4	грн/год	280	75	28

Графічні характеристики описаних вище параметрів (рисунок 4.2 - 4.5):

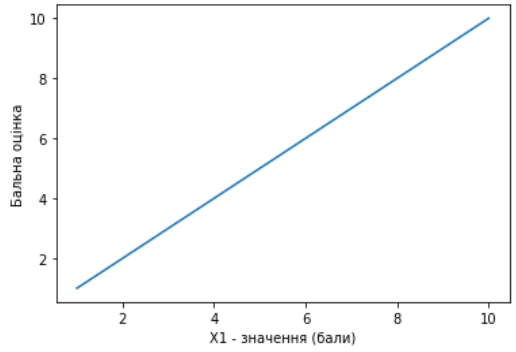


Рисунок 4.2 - X1

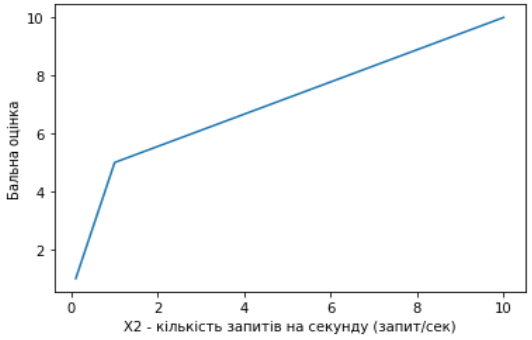


Рисунок 4.3 - X2

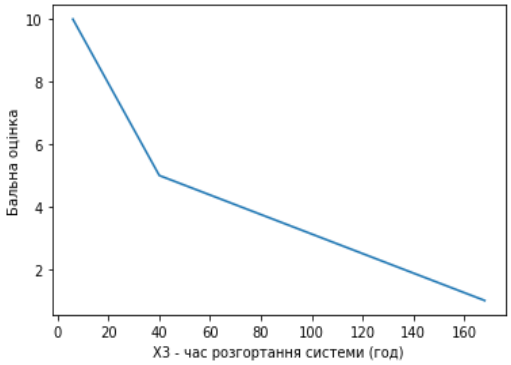


Рисунок 4.4 - X3

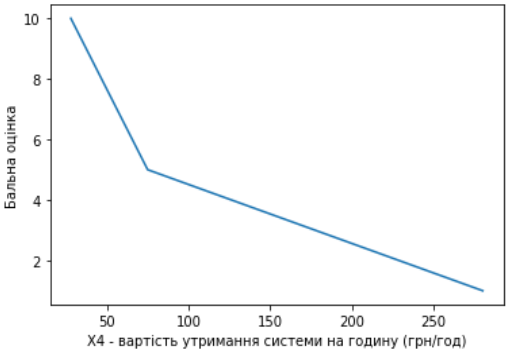


Рисунок 4.5 - X4

Вагомість параметрів визначається методом попарного їх порівняння на основі результатів ранжування експертами. Наведемо результати експертного ранжування (таблиця 4.3): — достовірне, виходячи із відповідної нерівності.

Таблиця 4.3

Позначення параметра	Одиниця вимірювання	R_1	R_2	R_3	R_4	R_5	R_6	R_7	Сума рангів R_i	Відхилення Δ_i	$\Delta^2{}_i$
X_1	бали	4	4	4	4	4	4	4	28	10.5	110.25
X_2	запити/сек	1	2	1	2	1	1	2	10	-7.5	56.25
X_3	год	3	3	2	3	2	3	3	19	1.5	2.25
X_4	грн/год	2	1	3	1	3	2	1	13	-4.5	20.25
Разом		10	10	10	10	10	10	10	70	0	189

Визначимо коефіцієнт узгодженості:

$W = \frac{12S}{N^2(n^3-n)} = \frac{12*189}{7^2(4^3-4)} \approx 0.771 > 0.67$ - достовірне, виходячи із даної нерівності

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4. Зауважимо, що ранг 1 відповідає найбільшій вагомості

Таблиця 4.4

Параметри	1	2	3	4	5	6	7	Кінцева оцінка	Числове значення
X_1 та X_2	<	<	<	<	<	<	<	<	0.5
X_1 та X_3	<	<	<	<	<	<	<	<	0.5
X_1 та X_4	<	<	<	<	<	<	<	<	0.5
X_2 та X_3	>	>	>	>	>	>	>	>	1.5
X_2 та X_4	>	<	>	<	>	>	<	>	1.5
X_3 та X_4	<	<	>	<	>	<	<	<	0.5

Таблиця 4.5

Параметри X_i	Параметри X_j				Перша ітерація		Друга ітерація		Третя ітерація	
	1	2	3	4	b_i	$K_{вi}$	b_i^1	$K_{вi}^1$	b_i^2	$K_{вi}^2$
X1	1	0.5	0.5	0.5	2.5	0.156	9.25	0.156	33.91	0.157
X2	1.5	1	1.5	1.5	5.5	0.344	21.25	0.361	77.87	0.361
X3	1.5	0.5	1	0.5	3.5	0.218	12.25	0.208	44.87	0.207
X4	1.5	0.5	1.5	1	4.5	0.282	16.25	0.275	59.45	0.275
Всього:					16	1	59	1	216	1

Відносні оцінки розраховуються декілька разів доти, доки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). Запишемо результати в таблицю 4.5. Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

4.3 Аналіз рівня якості варіантів реалізації функцій

Таблиця 4.6

Основні функції	Варіанти реалізацій	Параметри	Абсолютне Значення -	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	В	X_1	4	4	0.157	0.628
		X_3	30	4	0.207	0.828
F_2	Б	X_1	7	7	0.157	1.099
F_3	А	X_1	9	9	0.157	1.413
		X_2	0.5	2	0.361	0.722
		X_3	16	8	0.207	1.656
		X_4	62	6	0.275	1.65
	Б	X_1	3	3	0.157	0.471
		X_2	10	10	0.361	3.61
		X_3	40	5	0.207	1.035

		X_4	100	5	0.275	1.375

$K_1 = 0.628 + 0.828 + 1.099 + 1.413 + 0.722 + 1.656 + 1.65 = 7.996$

$K_2 = 0.628 + 0.828 + 1.099 + 0.471 + 3.61 + 1.035 + 1.375 = 9.046$

Отже, другий варіант, що передбачає розгортання системи в k8s кластері дає більший результат. Тому віддаємо йому перевагу.

4.4 Економічний аналіз варіантів розробки ПП

Обидва варіанти включають в себе три окремих етапи:

1. Підготовка даних
2. Дослідження й тренування нейронної мережі
3. Розгортання програмного продукту: для варіанту 1 - розгортання на віртуальній машині; для варіанту 2 - розгортання в k8s кластері

Для завдання 1 (Алгоритм складності 2, ступінь новизни В, вид використаної інформації НДІ)

$$T_p = 19,\ K_{\text{п}} = 0.72,\ K_{\text{СК}} = 1,\ K_{\text{СТ.М}} = 1.2$$

$$T_1 = 19*0.72*1*1.2 = 16.416 \text{ людино-днів}$$

Для завдання 2 (Алгоритм складності 1, ступінь новизни Б, вид використаної інформації НДІ)

$$T_p = 64,\ K_{\text{п}} = 1.021,\ K_{\text{СК}} = 1,\ K_{\text{СТ.М}} = 1.2$$

$$T_2 = 64*1.021*1*1.2 = 78.4128 \text{ людино-днів}$$

Для завдання 3 (при реалізації варіанту 1 - розгортання на віртуальній машині) (Алгоритм складності 2, ступінь новизни В, вид використаної інформації НДІ)

$$T_p = 27,\ K_{\text{п}} = 0.72,\ K_{\text{СК}} = 1, K_{\text{СТ.М}} = 1.2$$

$$T^1_3 = 27*0.72*1*1.2 = 23.328 \text{ людино-днів}$$

Для завдання 3 (при реалізації варіанту 2 - розгортання в k8s кластері), (Алгоритм складності 2, ступінь новизни Б, вид використаної інформації НДІ)

$$T_p = 27,\ K_{\text{п}} = 1.08,\ K_{\text{СК}} = 1, K_{\text{СТ.М}} = 1.2$$

$$T^1_3 = 27*1.08*1*1.2 = 34.992 \text{ людино-днів}$$

$$T_1 = (16.416 + 78.4128 + 23.328) * 8 = 945 \text{ людино-годин}$$

$$T_2 = (16.416 + 78.4128 + 34.992) * 8 = 1038 \text{ людино-годин}$$

В розробці та проведенні дослідження приймають участь 2 програмісти з окладом 28 000 грн

Зарплата розробників становить погодинно:

$$C = (28000+28000) / (2* 21 * 8) = 166.6 \text{ грн}$$

Зарплата поваріантно:

$$C_1 = 945*166.6 = 157437 \text{ грн}$$

$$C_2 = 1038*166.6 = 172930.8 \text{ грн}$$

Відрахування на соціальний внесок становить 22%:

$$C^v_1 = 0.22 * 157437 = 34636.14 \text{ грн}$$

$$C^v_2 = 0.22 * 172930.8 = 38044.78 \text{ грн}$$

Визначаємо витрати на оплату однієї машино-години. З урахуванням заробітної плати програміста в розмірі 28000 грн з коефіцієнтом зайнятості 0.2, маємо:

$$C_Г = 12*М* K_3 = 12 *28000* 0.2 = 67200 \text{ грн}$$

З урахуванням додаткової заробітної плати:

$$C_{ЗП} = C_Г * (1+ K_3) = 67200 * (1 + 0.2) = 80640 \text{ грн}$$

Відрахування на соціальний внесок:

$$C_{ВІД} = C_{ЗП} * 0.22 = 80640 * 0{,}22 = 17740.8 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн

$$C_А = K_{ТМ} * K_А * Ц_{ПР} = 1.15 * 0.25 * 8000 = 2300 \text{ грн.}$$

Витрати на ремонт та профілактику можна підрахувати:

$$C_Р = K_{ТМ} * Ц_{ПР} * K_Р = 1.15 * 8000 * 0.05 = 460 \text{ грн.,}$$

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) * t_3 * K_В = (365 – 104 – 11 – 16) * 8 * 0.9 = 1684.8$$

Тепер рахуємо витрати на оплату електроенергії (з урахуванням ПДВ):

Множником 1.2 враховуємо ПДВ

$$C_{ЕЛ} = Т_{ЕФ} * N_C * К_З * Ц_{ЕН} * 1.2 = 1684.8*0.3*0.6*1.46255*1.2 = 532.25 \text{ грн.}$$

Накладні витрати рахуються таким чином:

$$С_Н = Ц_{ПР} *0.67 = 8000 \cdot 0{,}67 = 5360 \text{ грн}$$

Річні експлуатаційні витрати:

$$С_{ЕКС} = 80640 + 17740.8 + 2300 + 460 + 532.25 + 5360 = 107033.05 \text{ грн}$$

Собівартість однієї машино-години ЕОМ становитиме:

$$С_{М-Г} = С_{ЕКС} / Т_{ЕФ} = 107033.05/1684.8 = 63.53 \text{ грн/год}$$

Витрати на оплату машинного часу складають в залежності від варіанту:

$$С^1_М = 63.53 * 945 = 60035.85 \text{ грн}$$

$$С^2_М = 63.53 * 1038 = 65944.14 \text{ грн}$$

Накладні витрати складають 67% від заробітної плати:

$$С^1_Н = 0.67 * 157437 = 105482.79 \text{ грн}$$

$$С^2_Н = 0.67 * 172930.8 = 115863.64 \text{ грн}$$

Таким чином, вартість розробки ПП та проведення дослідів складає:

$$С_1 = 157437 + 34636.14 + 60035.85 + 105482.79 = 357591.78 \text{ грн}$$

$$С_2 = 172930.8 + 38044.78 + 65944.14 + 115863.64 = 392783.36 \text{ грн}$$

Проведемо розрахунок техніко-економічного рівня:

$$К_{ТЕР1} = 7.996 / 357591.78 = 2.236 * 10^{-5}$$

$$К_{ТЕР2} = 9.046 / 392783.36 = 2.303 * 10^{-5}$$

Отже найбільш ефективним є другий варіант з коефіцієнтом техніко економічного рівня $2.303 * 10^{-5}$, що передбачає використанння Kubernetes кластеру. Така конфігурація є реалізуємою в технічному плані та надає достатньо якісні результати.

ВИСНОВОК

У даній роботі розглянуто основні типи глибоких нейронних мереж, які наразі активно застосовують до задач обробки природної мови та їх основні складові та механізми, розглянуто різні задачі та методи трансферного навчання в цілому та конкретні види підходів для напрямку обробки природної мови.

Також запропоновано формалізацію абстрактної задачі та загальний підхід до її вирішення через вирішення задачі розмітки послідовності та відкидання хибно-позитивних кандидатів. Було розглянуто приклад конкретної задачі та описано вибірки даних, які використовувалися для її вирішення. Для експериментальних досліджень було описано метрику та стратегію оцінки якості результатів роботи моделей, показано шлях виведення оцінок параметрів моделей за допомогою методу максимальної правдоподібності. Власне експерименти було проведено для двох типів моделей: з LSTM кодувальником та BERT кодувальником для трьох підходів: базового, трансферного навчання з адаптацією предметної області й задачі; трансферного навчання з адаптацією задачі; та трансферного навчання з адаптацією задачі з відкиданням хибно-позитивних кандидатів. За допомогою останнього підходу отримано найкращий результат.

Зрештою розглянули особливості архітектури та імплементації системи та описали технології, які були використані для створення та розгортання системи. Фінальна версія представляє собою веб систему мікросервісної архітектури з управляючим сервісом, кроулером та сервісом моделі машинного навчання, яка може бути розгорнута на віртуальній машині чи в кластері.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hochreiter S. Long short-term memory / Sepp Hochreiter, Jürgen Schmidhub // Neural Computation. — 1997. — № 9. — С. 1735—1780.
2. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions / Sepp Hochreiter // International Journal of Uncertainty Fuzziness and Knowledge-Based Systems. — 1998. — № 6. — С. 107—116.
3. A Comprehensive Survey on Transfer Learning [Електронний ресурс] / Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, Qing He // Веб-сайт arxiv.org — 2019. — С. 27. — Режим доступу до ресурсу: https://arxiv.org/pdf/1911.02685.pdf
4. Yuan-Pin L. Improving EEG-Based Emotion Classification Using Conditional Transfer Learning [Електронний ресурс] / Lin Yuan-Pin, Jung Tzyy-Ping // Веб-сайт arxiv.org — 2017. — С. 22. — Режим доступу до ресурсу: https://arxiv.org/pdf/2001.11337.pdf
5. Malte A. A Comprehensive Survey on Transfer Learning [Електронний ресурс] / Aditya Malte, Pratik Ratadiya // Веб-сайт arxiv.org — 2017. — С. 29. — Режим доступу до ресурсу: https://arxiv.org/pdf/1451.0045.pdf
6. Attention Is All You Need / Google Inc. — К. : Conference on Neural Information Processing Systems, 2017. — 15 с.
7. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / Google Inc. — К. : NAACL, 2019. — 16 с.
8. Офіційний репозиторій моделей BERT від Google Inc. [Електронний ресурс] / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova // Веб-сайт github.com — 2018. — Режим доступу до ресурсу: https://github.com/google-research/bert

9. Датасет CoNLL-2003 Language-Independent Named Entity Recognition [Електронний ресурс] / Linguistics department of the faculty of Arts of the University of Antwerp. — 2003. — Режим доступу до ресурсу: https://www.clips.uantwerpen.be/conll2003/ner/
10. Ramshaw L. A. Text Chunking using Transformation-Based Learning / Lance A. Ramshaw, Mitchell P. Marcus — Third Workshop on Very Large Corpora, 1995. — С. 13.
11. Contextual String Embeddings for Sequence Labeling / Humboldt University of Berlin — К. : COLING, 2018. — 12 с.
12. Офіційний репозиторій моделей FLAIR [Електронний ресурс] / Alan Akbik, Duncan Blythe, Roland Vollgraf // Веб-сайт github.com — 2018. — Режим доступу до ресурсу: https://github.com/flairNLP/flair
13. Kingma D. P. Adam: A Method for Stochastic Optimization / Diederik P. Kingma, Jimmy Ba. — К. : CoRR, 2015. — 15 с.

ДОДАТОК А ЛІСТІНГ ПРОГРАМНОГО ПРОДУКТУ

Тренування мережі на базі LSTM

```
from flair.data import Corpus
from flair.embeddings import TokenEmbeddings, WordEmbeddings, StackedEmbeddings,
PooledFlairEmbeddings, FlairEmbeddings
from typing import List
from flair.datasets import ColumnCorpus
from flair.models import SequenceTagger
import click

columns = {0: 'text', 1: 'pos', 2: 'chunk', 3: 'ner'}
data_folder = './'
corpus: Corpus = ColumnCorpus(data_folder, columns,
                              train_file='flair_org_addr_dataset_train',
                              test_file='flair_org_addr_dataset_test',
                              dev_file='flair_org_addr_dataset_dev', in_memory=False)

tag_type = 'ner'
tag_dictionary = corpus.make_tag_dictionary(tag_type=tag_type)
model_output_folder = 'flair_full_org_addr_fw_bw'

embedding_types: List[TokenEmbeddings] = [

    # GloVe embeddings
    WordEmbeddings('glove'),

    # contextual string embeddings, forward
    FlairEmbeddings('news-forward-fast'),

    # contextual string embeddings, backward
    FlairEmbeddings('news-backward-fast')
  ]


embeddings: StackedEmbeddings = StackedEmbeddings(embeddings=embedding_types)

  # initialize sequence tagger

tagger: SequenceTagger = SequenceTagger(hidden_size=256,
                                        embeddings=embeddings,
                                        tag_dictionary=tag_dictionary,
                                        tag_type=tag_type,
                                        use_crf=True)


  # initialize trainer
from flair.trainers import ModelTrainer
```

```
trainer: ModelTrainer = ModelTrainer(tagger, corpus)

trainer.train(model_output_folder,
              learning_rate=0.1,
              mini_batch_size=32,
              max_epochs=15,
              shuffle=True,
              embeddings_storage_mode='none',
              checkpoint=True,
              anneal_with_restarts=True,
              num_workers=6,
              patience=1)
```

Тренування мережі на базі BERT

```
"""BERT NER Inference."""

from __future__ import absolute_import, division, print_function

import json
import os

import torch
import torch.nn.functional as F
from nltk import word_tokenize
from pytorch_transformers import (BertConfig, BertForTokenClassification,
                                  BertTokenizer)


class BertNer(BertForTokenClassification):

    def forward(self, input_ids, token_type_ids=None, attention_mask=None, valid_ids=None):
        sequence_output = self.bert(input_ids, token_type_ids, attention_mask, head_mask=None)[0]
        batch_size,max_len,feat_dim = sequence_output.shape
        valid_output = torch.zeros(batch_size,max_len,feat_dim,dtype=torch.float32,device='cuda' if
torch.cuda.is_available() else 'cpu')
        for i in range(batch_size):
            jj = -1
            for j in range(max_len):
                    if valid_ids[i][j].item() == 1:
                        jj += 1
                        valid_output[i][jj] = sequence_output[i][j]
        sequence_output = self.dropout(valid_output)
        logits = self.classifier(sequence_output)
        return logits

class Ner:
```

```
    def __init__(self,model_dir: str):
        self.model , self.tokenizer, self.model_config = self.load_model(model_dir)
        self.label_map = self.model_config["label_map"]
        self.max_seq_length = self.model_config["max_seq_length"]
        self.label_map = {int(k):v for k,v in self.label_map.items()}
        self.device = "cuda" if torch.cuda.is_available() else "cpu"
        self.model = self.model.to(self.device)
        self.model.eval()

    def load_model(self, model_dir: str, model_config: str = "model_config.json"):
        model_config = os.path.join(model_dir,model_config)
        model_config = json.load(open(model_config))
        model = BertNer.from_pretrained(model_dir)
        tokenizer = BertTokenizer.from_pretrained(model_dir,
do_lower_case=model_config["do_lower"])
        return model, tokenizer, model_config

    def tokenize(self, text: str):
        """ tokenize input"""
        words = word_tokenize(text)
        tokens = []
        valid_positions = []
        for i,word in enumerate(words):
            token = self.tokenizer.tokenize(word)
            tokens.extend(token)
            for i in range(len(token)):
                if i == 0:
                    valid_positions.append(1)
                else:
                    valid_positions.append(0)
        return tokens, valid_positions

    def preprocess(self, text: str):
        """ preprocess """
        tokens, valid_positions = self.tokenize(text)
        ## insert "[CLS]"
        tokens.insert(0,"[CLS]")
        valid_positions.insert(0,1)
        ## insert "[SEP]"
        tokens.append("[SEP]")
        valid_positions.append(1)
        segment_ids = []
        for i in range(len(tokens)):
            segment_ids.append(0)
        input_ids = self.tokenizer.convert_tokens_to_ids(tokens)
        input_mask = [1] * len(input_ids)
        while len(input_ids) < self.max_seq_length:
            input_ids.append(0)
            input_mask.append(0)
            segment_ids.append(0)
```

```
            valid_positions.append(0)
        return input_ids,input_mask,segment_ids,valid_positions

    def predict(self, text: str):
        input_ids,input_mask,segment_ids,valid_ids = self.preprocess(text)
        input_ids = torch.tensor([input_ids],dtype=torch.long,device=self.device)
        input_mask = torch.tensor([input_mask],dtype=torch.long,device=self.device)
        segment_ids = torch.tensor([segment_ids],dtype=torch.long,device=self.device)
        valid_ids = torch.tensor([valid_ids],dtype=torch.long,device=self.device)
        with torch.no_grad():
            logits = self.model(input_ids, segment_ids, input_mask,valid_ids)
        logits = F.softmax(logits,dim=2)
        logits_label = torch.argmax(logits,dim=2)
        logits_label = logits_label.detach().cpu().numpy().tolist()[0]

        logits_confidence = [values[label].item() for values,label in zip(logits[0],logits_label)]

        logits = []
        pos = 0
        for index,mask in enumerate(valid_ids[0]):
            if index == 0:
                continue
            if mask == 1:
                logits.append((logits_label[index-pos],logits_confidence[index-pos]))
            else:
                pos += 1
        logits.pop()

        labels = [(self.label_map[label],confidence) for label,confidence in logits]
        words = word_tokenize(text)
        assert len(labels) == len(words)
        output = [{"word":word,"tag":label,"confidence":confidence} for word,(label,confidence) in
zip(words,labels)]
        return output
```

Код веб сервісу

```
from flask import Flask, jsonify, request
import html2text
import requests

from extractors.extractor import Extractor
from tools.ner.prepare_NER_settings import
TAGGER_POLICY_PAGE_TORCH_MODEL_FILE,
SELECTOR_POLICY_PAGE_FLAIR_MODEL_FILE
from server_common import InvalidUsage
import logging
import logging.handlers
```

```
from utils.common_logging import GZipRotator

app = Flask(__name__)
app.config.from_object('policy_page_server_settings')


LOG_FILE = 'policy_page_server.log'

@app.errorhandler(InvalidUsage)
def handle_invalid_usage(error):
    response = jsonify(error.to_dict())
    response.status_code = error.status_code
    return response

def get_logger():
    logformatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
    log = logging.handlers.TimedRotatingFileHandler(LOG_FILE, 'midnight', 1, backupCount=14)
    log.setLevel(logging.INFO)
    log.setFormatter(logformatter)
    log.rotator = GZipRotator()

    logger = logging.getLogger('PolicyPageServer')
    logger.addHandler(log)
    logger.setLevel(logging.INFO)

    return logger

logger = get_logger()

extractor = Extractor(TAGGER_POLICY_PAGE_TORCH_MODEL_FILE,
SELECTOR_POLICY_PAGE_FLAIR_MODEL_FILE)

@app.route('/api/process_text', methods=['POST', 'GET'])
def api_process_text():
    try:
        sample = request.get_json()
        print(sample)
        text = sample["text"] if 'text' in sample and sample['text'] else ''
        policy_url = sample['policy_url'] if 'policy_url' in sample and sample['policy_url'] else ''
        return jsonify(extractor.extract(text, policy_url))
    except Exception as e:
        return e

@app.route('/api/process_url', methods=['POST', 'GET'])
def api_process_url():
    try:
        sample = request.get_json()
        print(sample)
        sample["text"] = html2text.html2text(requests.get(sample['url']).text)
        text = sample["text"] if 'text' in sample and sample['text'] else ''
```

```
        policy_url = sample['url'] if 'url' in sample and sample['url'] else ''
        return jsonify(extractor.extract(text, policy_url))
    except Exception as e:
        return e


@app.route('/', methods=['POST'])
def ping():
    return 'pong'


if __name__ == "__main__":
    app.run(debug=app.config["DEBUG"],
            host=app.config["HOST"],
            port=app.config["PORT"])
```

ДОДАТОК Б ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДЛЯ ДОПОВІДІ

Алгоритм видобутку інформації з неструктурованих текстових джерел

Автор: студент 4-го курсу
групи КА-61
Баздирев А. А.

Актуальність

На сьогоднішні день, в еру діджиталізації та стрімкого розвитку мережевих технологій дуже важливо своєчасно отримувати максимально повну інформацію в структурованому вигляді. Алгоритм видобутку інформації з неструктурованих джерел та методологія його побудови може знайти широке застосування для компаній, що займаються стратегічним консультуванням, аналізом ризиків та створенням загальних всеохоплюючих комерційних баз знань та експертних систем.

Постановка задачі

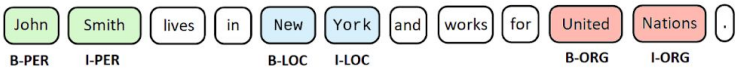
1. Побудувати загальний підхід до видобутку інформації з неструктурованих текстових джерел.
2. Застосувати підхід до реальної задачі про видобуток офіційних імен та адрес з розбиттям на окремі компоненти з сайтів компаній.
3. Дослідити вплив застосування різноманітних технік трансферного навчання для реальної задачі.
4. Створити та розгорнути систему на основі мікросервісної архітектури для вирішення вищеописаної задачі.

Методи, предмет та об'єкт дослідження дослідження

- Об'єкт дослідження - неструктуровані текстові дані, зокрема тексти з веб-сайтів
- Предмет дослідження - рекурентні нейронні мережі, нейронні мережі трансформери
- Методи дослідження - метод максимальної правдоподібності, методи трансферного навчання, зокрема перенесення задачі та предметної області

Задача розмітки послідовності

Нехай є деякий скінченний словник токенів V = {w0, w1,..wn}, та простір цільової змінної (можливі класи) Y. Задача розмітки послідовності полягає в тому, щоб побудувати модель, щоб для довільної послідовність T = {t0, t1,..tm}, при чому i=1,m; ti є V оцінити таку величину: Pi(y=yj | t0, t1,..tm; Θ) - ймовірність, що i-й токен належить до класу yj за умови послідовності токенів t0, t1,..tm ; Θ - параметри моделі. Параметри моделі будемо оцінювати за методом максимальної правдоподібності.



Задача відкидання хибно-позитивних кандидатів

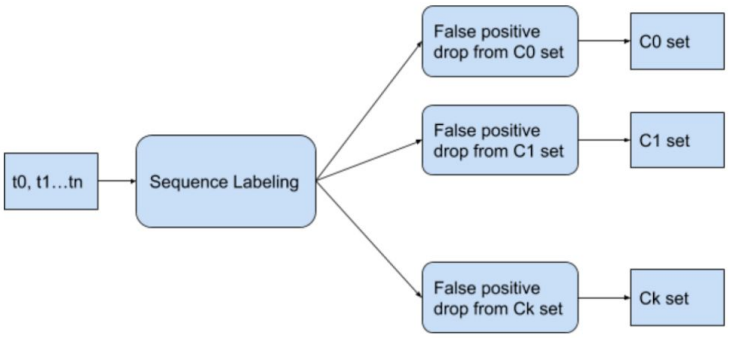
X = {(t, p, d)} - простір факторів, що складається з таких трійок: t - токен, p - розподіл ймовірностей для t від деякої моделі розмітки послідовностей, d - вектор додаткових мета-факторів. Зауважимо, що якщо не задані фактори d, то подальша побудова моделі не має сенсу, оскільки реально не додається ніяких додаткових факторів, тому вважатимемо, що фактори d не є тривіальними.

Y = {0, 1} - простір цільової змінної, де 0 - відповідає хибно-позитивному токену, а 1 - істинно-позитивному.

Параметри моделі будемо оцінювати за методом максимальної правдоподібності.

Загальний підхід

Нехай {c0,..ck} задані класи, x = [t0, t1,..tn] - документ (послідовність слів)



Тренувальна вибірка

Введите те

Children's Personal Information

We don't provide AWS Offerings for purchase by children. If you're u with the involvement of a parent or guardian.

Retention of Personal Information

We keep your personal information to enable your continued use of in order to fulfill the relevant purposes described in this Privacy Noti tax and accounting purposes), or as otherwise communicated to you information varies depending on the purpose for its use, and we will accordance with applicable law.

Contacts, Notices, and Revisions

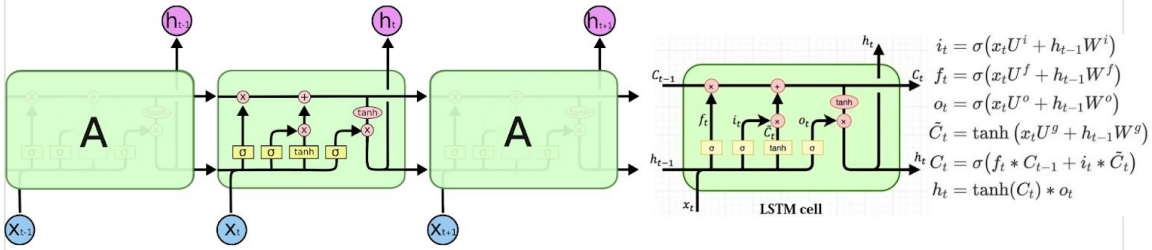
If you have any concern about privacy at AWS or want to contact one of our data controllers, please contact us with a thorough description, and we will try to resolve it. You may also contact us at the addresses below:

- For any prospective or current customers of Amazon Web Services, Inc., our mailing address is: Amazon Web Services, Inc., 410 Terry Avenue North, Seattle, WA 98109-5210, ATTN: AWS Legal


```
{
  "address": {
    "ADDRESS1": "410 Terry Avenue North",
    "ADDRESS2": null,
    "CITY": "Seattle",
    "STATE": "WA",
    "ZIP": "98109-5210",
    "COUNTRY": null
  },
  "organization": "Amazon Web Services, Inc."
}
```

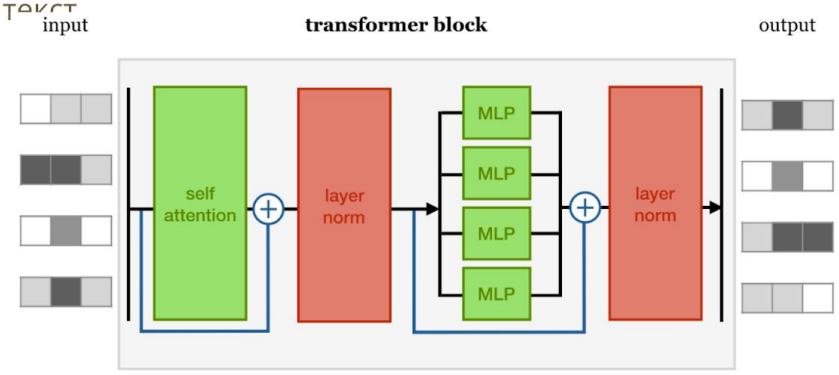
LSTM

Введите текст



Transformer block

Введите текст



Імплементація моделі розмітки послідовності

Позначимо функцію кодувальника як E (в нашому випадку LSTM/BERT).

E: {x1, x2,..xm} -> {y1, y2,..ym} - відображення вхідної векторної послідовності {x1, x2,..xm} в результат роботи кодувальника - вихідну векторну послідовність {y1, y2,..ym}. Щоб для виходу кодувальника yi оцінити розподіл ймовірностей належності до класів {1, 2, .. K} застосуємо до виходів лінійний оператор W (також належить до параметрів моделі), що діє з простору такої ж розмірності, що й yi в простір розмірності K, та застосуємо до Wyi функцію softmax.

$$softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Метрика

В якості основної метрики будемо використовувати MacroF1

$$MacroF1 = 2 * \frac{MacroPrecision * MacroRecall}{MacroPrecision + MacroRecall}$$

$$MacroPrecision = \frac{1}{K}\sum_{i=1}^{K} Precision_i; \quad MacroRecall = \frac{1}{K}\sum_{i=1}^{K} Recall_i$$

$$\text{Precision}_i = \frac{TruePositives_i}{TruePositives_i + FalsePositives_i} \qquad \text{Recall}_i = \frac{TruePositives_i}{TruePositives_i + FalseNegatives_i}$$

Базовий підхід

Базовий підхід полягає в тому, що би виконувати тренування для моделей на нашій вибірці починаючи “з нуля” від випадкової ініціалізації вагів.

Модель	середнє MacroF1	с.кв. відх. MacroF1
LSTM	0.6770	0.0116
BERT	0.6642	0.0184

Підхід трансферного навчання: адаптація предметної області та задачі

Цей підхід полягає в тому, що б дотренувати для нашої вибірки та задачі модель, яку вже попередньо тренували на великому обсягу даних для деякої іншої задачі.

Модель	середнє MacroF1	с.кв. відх. MacroF1
LSTM	0.6941	0.0072
BERT	0.7522	0.0089

Підхід трансферного навчання: адаптація задачі

Цей підхід полягає в тому, що б зробити попереднє тренування моделей з попереднього підходу спочатку на даних з релевантної предметної області (можливо для іншої задачі), і тільки після того тренувати для нашої вибірки і задачі.

Модель	середнє MacroF1	с.кв. відх. MacroF1
LSTM	0.7549	0.0048
BERT	0.8204	0.0032

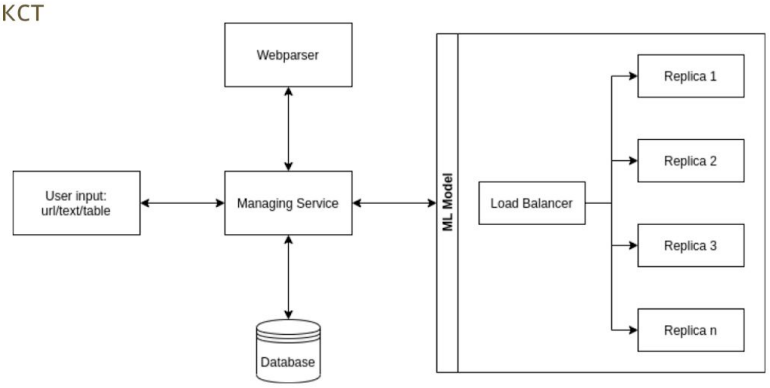
Відкидання хибно-позитивних кандидатів

Будемо відкидати хибно-позитивні кандидати для ORG класу. Цей підхід показує найкращий результат за метрикою.

Модель	F1 для ORG класу	середнє MacroF1	с.кв. відх. MacroF1
BERT	0.6502	0.8204	0.0032
BERT + FP drop	0.8423	0.8489	0.0030

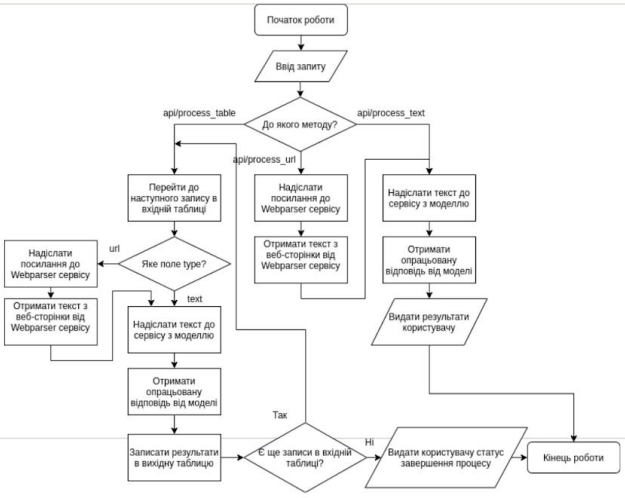
Розгортання системи в кластері

Введите текст



Робота управляючого сервісу

Введите текст



Демонстрація роботи програми

Thank you for visiting this website, which is operated by an Affiliate of Sinclair Television Group, Inc. ("Sinclair"). This site is one of a network of ad-supported sites operated by Affiliates of Sinclair each of which also operates a local television station (each a "Sinclair Affiliate Site" and, collectively, the "Sinclair Network of Sites"). Each Sinclair Affiliate Site has adopted this privacy statement to the extent applicable. "Affiliate" means a company controlling, controlled by or under common control with another company, or a company which shares common management.

Notice of Copyright Infringement. If you are an owner of intellectual property who believes your intellectual property has been improperly posted or distributed via this Service, please notify us through our feedback procedure or by sending a notice by U.S. Mail to Sinclair Broadcast Group, Inc., 10706 Beaver Dam Road, Hunt Valley, MD 21030 Attn: Legal Department. Your notice to us must include the following information: (1) a physical or electronic signature of a person authorized to act on behalf of the owner of the copyrighted work allegedly infringed; (2) a description of the copyrighted work or works that allegedly have been infringed; (3) a description of where on the Service the allegedly infringing material appears that will allow us to locate the material; (4) a statement by you that you have a good faith belief that the allegedly infringing use has not been authorized by the copyright owner, its agent, or the law; and (5) a statement by you that the information in your notice is accurate, and under penalty of perjury, that you are authorized to act on behalf of the owner of the copyrighted work that has allegedly been infringed.

```
[1]: import json
     import requests

     service_url = 'http://a35199bad72d211eab5620ee4f799c0f-1681897105.us-west-2.elb.amazonaws.com/api/process_url'
     resp = requests.post(service_url, json={'url': 'http://www.foxnebraska.com/global/story.asp?s=18991'}).text
     resp = json.loads(resp)
     resp

[1]: {'ORG': [{'text': 'Sinclair Television Group, Inc.', 'rank': 0.930471},
       {'text': 'Sinclair Broadcast Group, Inc.', 'rank': 0.610294}],
      'ADDRESS': [{'ADDR1': '10706 Beaver Dam Road',
        'ADDR2': None,
        'CITY': 'Hunt Valley',
        'STATE': 'MD',
        'ZIP': '21030',
        'COUNTRY': None,
        'rank': 0.851902}]}
```

Висновки

- Побудовано загальний підхід до видобутку інформації з неструктурованих текстових джерел
- Підхід протестовано на реальній задачі
- Досліджено вплив технік трансферного навчання на якість роботи моделей
- Фінальну систему розгорнуто в хмарі

Шляхи подальшого розвитку

- Збір багатомовного корпусу з різних предметних областей для запропонованої задачі
- Тестування загального підходу для інших задач
- Дослідження мовно-незалежних підходів трансферного навчання

Дякую за увагу!